Grado en Ingeniería Informática

Trabajo Final de Grado

Gestión de entorno de virtualización para la creación automática de máquinas virtuales

---

Autor:
Santiago CONESA BONO

Tutor académico:
Carlos Antonio HERNÁNDEZ ESPINOSA

Supervisor:
Robert ARTERO SANCHO

Fecha de lectura: 13 de noviembre del 2014
Curso académico 2013/2014

Agradecimientos

Durante la elaboración de este proyecto han participado, directa o indirectamente, una serie de personas a las que quiero dar las gracias.

En primer lugar, al tutor de este proyecto, Carlos Antonio Hernández Espinosa, por su implicación y disponibilidad durante el desarrollo del proyecto.

En segundo lugar, al supervisor de este proyecto, Robert Artero Sancho, por su implicación y ayuda recibida durante el desarrollo de la aplicación.

Por supuesto, agradecer también a Miquel Ramon Ortega i Tido y Ivan Barrichina Bellmunt, integrantes de la empresa Sofistic Telematic Security S.L., por su ayuda en todo momento y sus consejos.

Por último, agradecer a mi familia y amigos su apoyo en los momentos difíciles desde que empezara a estudiar esta bonita pero dura carrera como es el Grado en Ingeniería Informática, y que siempre me han sabido inyectar moral en mis peores momentos.

Resumen

En este documento se presenta el resultado del Trabajo Final de Grado y estancia en prácticas consistente en un aplicación web, capaz de crear y gestionar máquinas virtuales, desarrollada en la empresa *Sofistic Telematic Security*.

La funcionalidad ofrecida por esta aplicación es que permite crear máquinas virtuales vps en un servidor web.

La aplicación es capaz de gestionar las máquinas virtuales, es decir, cambiar su hardware y software de manera automática dependiendo de la utilidad que el usuario le quiere dar a cada máquina.

Para llevar a cabo el proyecto ha sido necesario seguir las diferentes fases que lleva la creación de software, incluyendo el análisis de requisitos, la gestión del alcance y restricciones, la planificación del diseño y la implementación.

Palabras clave

Máquina virtual VPS.

Aplicación Web.

Servidor Web.

Estadísticas de rendimiento.

Keywords

Virtual machine VPS.

Web application.

Web server.

Performance statistics.

Índice

Índice de figuras

Índice de tablas

1 Introducción

Este documento incluye la memoria del proyecto realizado durante la estancia de prácticas en la empresa Sofistic (Sofistic Telematic Security S.L) [1]. Dicha empresa está ubicada en el edificio denominado Espaitec 2 de la Universidad Jaume I de Castellón. Desde 2009, Sofistic esta especializada en la seguridad informática, con una serie de productos y servicios entorno a soluciones de eCommerce, Identity Management, y gestión de sistemas, con profesionales y especialistas que han implantado, desplegado y auditado soluciones seguras a grandes marcas nacionales.

1.1 Historia

El concepto fundamental de la entrega de los recursos informáticos a través de una red global tiene sus raíces en los años sesenta. La idea de una "red de computadoras intergaláctica" fue introducida en los años sesenta por JCR Licklider, quien era responsable de permitir el desarrollo de ARPANET (Advanced Research Projects Agency Network) en 1969 [2]. Su visión era que todo el mundo pudiese estar interconectado y poder acceder a los programas y datos desde cualquier lugar, explicó Margaret Lewis, directora de marketing de producto de AMD. "Es una visión que se parece mucho a lo que llamamos cloud computing".

Otros expertos atribuyen el concepto científico de la computación en nube a John McCarthy, quien propuso la idea de la computación como un servicio público, de forma similar a las empresas de servicios que se remontan a los años sesenta. John McCarthy, (1960): "Algún día la computación podrá ser organizada como un servicio público"[3].

Desde los años sesenta, la computación en nube se ha desarrollado a lo largo de una serie de líneas. La Web 2.0 es la evolución más reciente. Sin embargo, como Internet no empezó a ofrecer ancho de banda significativo hasta los años noventa, la computación en la nube ha sufrido algo así como un desarrollo tardío. Uno de los primeros hitos de la computación en nube es la llegada de Salesforce.com en 1999, que fue pionero en el concepto de la entrega de aplicaciones empresariales a través de una página web simple. La firma de servicios allanó el camino para que tanto especialistas como empresas tradicionales de software pudiesen publicar sus aplicaciones a través de Internet.

El siguiente desarrollo fue Amazon Web Services en 2002, que prevé un conjunto de servicios basados en la nube, incluyendo almacenamiento, computación e incluso la inteligencia humana a través del Amazon Mechanical Turk. Posteriormente en 2006, Amazon lanzó su Elastic Compute Cloud (EC2) como un servicio comercial que permite a las pequeñas empresas y los particulares alquilar equipos en los que se ejecuten sus propias aplicaciones informáticas. 8 "Amazon EC2/S3 fue el que ofreció primero servicios de infraestructura en la nube totalmente accesibles", dijo Jeremy Allaire, CEO de Brightcove, que proporciona su plataforma SaaS de vídeo en línea a las estaciones de televisión de Reino Unido y periódicos. George Gilder, 2006: "El PC de escritorio está muerto. Bienvenido a la nube de Internet, donde un número enorme de instalaciones a lo largo de todo el planeta almacenarán todos los datos que usted podrá usar alguna vez en su vida".

Otro hito importante se produjo en 2009, cuando Google entre otros, empezaron a ofrecer aplicaciones basadas en navegador. Servicios, como Google Apps. "La contribución más importante a la computación en nube ha sido la aparición de "aplicaciones asesinas" de los gigantes de tecnología como Microsoft y Google. Cuando dichas compañías llevan a cabo sus servicios de una manera que resulta segura y sencilla para el consumidor, el efecto 'pasar la pelota' en sí, crea un sentimiento de mayor aceptación de los servicios online", dijo Dan Germain, jefe de la oficina de tecnología en IT proveedor de servicios Cobweb Solutions.

Otro de los factores clave que han permitido evolucionar a la computación en la nube según el británico y pionero en computación en la nube Jamie Turner, han sido la tecnologías de virtualización, el desarrollo universal de alta velocidad de ancho de banda, y normas universales de interoperabilidad de software. Turner añadió: "A medida que la computación en nube se extiende, su alcance va más allá de un puñado de usuarios de Google Docs. Sólo podemos empezar a imaginar su ámbito de aplicación y alcance. Casi cualquier cosa puede ser utilizado en la nube".

1.2 Breve marco teórico

En los últimos años las empresas tecnológicas han empezado apostar por la computación en la nube [14].

La computación en la nube son servidores desde Internet encargados de atender las peticiones en cualquier momento. Se puede tener acceso a su información o servicio, mediante una conexión a Internet desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar. Esta medida reduce los costes, garantiza un mejor tiempo de actividad y que los sitios web sean invulnerables a los hackers, a los gobiernos locales y a sus redadas policiales.

“Cloud computing” o computación en la nube es un nuevo modelo de prestación de servicios de negocio y tecnología, que permite incluso al usuario acceder a un catálogo de servicios estandarizados y responder con ellos a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado, o incluso gratuitamente en caso de proveedores que se financian mediante publicidad o de organizaciones sin ánimo de lucro [13].

1.3 Conceptos

Para entender correctamente el proyecto y saber en que nos inspiremos para desarrollarlo, tenemos que entender una serie de conceptos.

1.3.1 Virtualización

Hablar de virtualización es hablar de un concepto que describe la posibilidad de tener varios sistemas operativos funcionando al mismo tiempo en un mismo equipo físico. Hace referencia a la capa de software encargada de generar una abstracción entre los recursos de una máquina (hardware), con el sistema operativo huésped, donde se compartirán dichos recursos en uno o más entornos de ejecución (equipos virtuales). Dicha capa de software se encarga de administrar los recursos principales de una computadora (CPU, Memoria, Red, Almacenamiento) y repartir los mismos a las máquinas correspondientes con el fin de tener varios equipos virtuales ejecutándose sobre el mismo equipo físico.

1.3.2 Maquina Virtual

Una máquina virtual, es un contenedor de software perfectamente aislado en el cual se puede ejecutar, de igual forma, tanto sistemas operativos como aplicaciones, como si se tratara de un equipo físico, ya que se comporta exactamente igual a como lo hace un equipo físico y contiene sus propios dispositivos, como: CPU, RAM, disco duro y tarjetas de interfaz de red (NIC) virtuales.

De igual manera, una maquina virtual trabaja de forma transparente para los diversos sistemas operativos para los que tiene soporte y de igual forma para las aplicaciones, ya que no hay establecida una diferencia entre una máquina virtual y una máquina física; lo mismo pasa con otros equipos de una red, aunque se componga exclusivamente de software y no está formada en ninguna de sus partes por algún componente de hardware, más que el equipo que las contiene.

Una maquina virtual trae consigo muchas ventajas, de las que podemos mencionar:

a) Compatibilidad.
b) Aislamiento.
c) Encapsulamiento.
d) Independencia de hardware.

Al combinar las características antes mencionadas, proporciona la libertad por ejemplo, para mover una máquina virtual de un equipo a otro, sin necesidad de efectuar ningún cambio en los controladores de dispositivo, en el sistema operativo o en sus respectivas aplicaciones, inclusive si existe espacio físico para el dispositivo en el centro de datos.

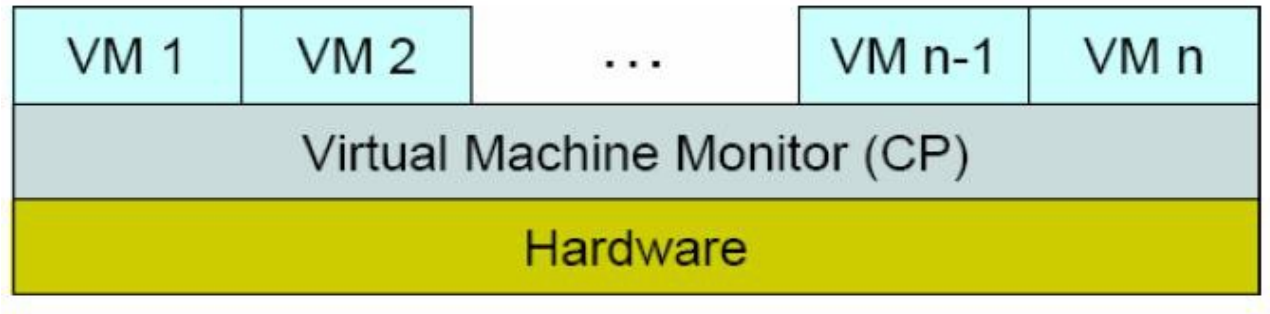


Figura 1: Arquitectura típica de una máquina virtual.

1.3.3 Servidor virtual

Se conoce como servidor virtual a una partición dentro de un servidor que habilita varias máquinas virtuales dentro de dicha máquina por medio de varias tecnologías.

Los servidores dedicados virtuales (SDV) usan una avanzada tecnología de virtualización, que le permite proveer acceso [root] y la capacidad de reiniciarlo cuando desee, igual que un servidor dedicado. Con la posibilidad de instalar sus propias aplicaciones y controlar completamente la configuración de su servidor, los SDV representan una alternativa económica y eficiente para aquellos que desean disfrutar los beneficios de un servidor dedicado pero aun no poseen el presupuesto para hacerlo.

1.4 Objetivos del proyecto

Una vez analizado el contexto de trabajo pasamos a explicar los objetivos que se pretenden conseguir con la aplicación que hemos desarrollado en este proyecto.

El objetivo principal que tiene el proyecto es el de desarrollar un producto fácil de utilizar e

intuitivo para los clientes que tengan que contratar o gestionar sus servidores virtuales. De esa manera, la empresa que lanza el producto, no tiene que estar pendiente de gestionar los servidores virtuales de sus clientes, ya que como hemos mencionado anteriormente, ellos serán capaces de gestionarlos a través del panel que se ha creado. Una característica del panel, es que todo esta bastante explicado, para que de esa forma, un usuario antes de presionar un botón, sepa en todo lugar que le sucederá al servidor virtual sobre el cual esta aplicando la acción. Más adelante, explicaremos con más detalle la interfaz gráfica.

También se intenta conseguir con la aplicación que los clientes puedan realizar un seguimiento de los recursos de cada máquina virtual. De esta forma, se habilita en el panel, unas opciones para ampliar o disminuir los recursos de las máquinas. Con esto se intenta conseguir que los clientes paguen en función de lo que necesitan en cada momento. Entonces, para la aplicación nos será necesario realizar un sistema de pago automático para no estar solicitando los datos al cliente cada vez que cambie un recurso o cada vez que se termine el periodo de contrato de una máquina.

Además, también se irán recopilando datos sobre las máquinas para que los usuarios sepan de manera clara y concisa, el trabajo que está realizando cada maquina. Así como también cuales son los recursos que mas explota y cuales son las horas en las cuales tiene que realizar un mayor esfuerzo para satisfacer las peticiones de los clientes.

En definitiva, lo que se pretende conseguir con la aplicación es facilitar la gestión de los servidores virtuales a los usuarios y entrar a competir en un mercado que día a día va creciendo, ya que está considerado como la tecnología del futuro.

2 Requisitos

En este apartado se definen detalladamente los requisitos de la aplicación. Se incluyen los casos de uso, los requisitos de datos y la plataforma tecnológica necesaria.

2.1 Casos de uso

Para dar una idea general de los requisitos del proyecto a continuación se mostrará un diagrama de casos de uso en el que se analizará la interacción que realiza cada actor del sistema.

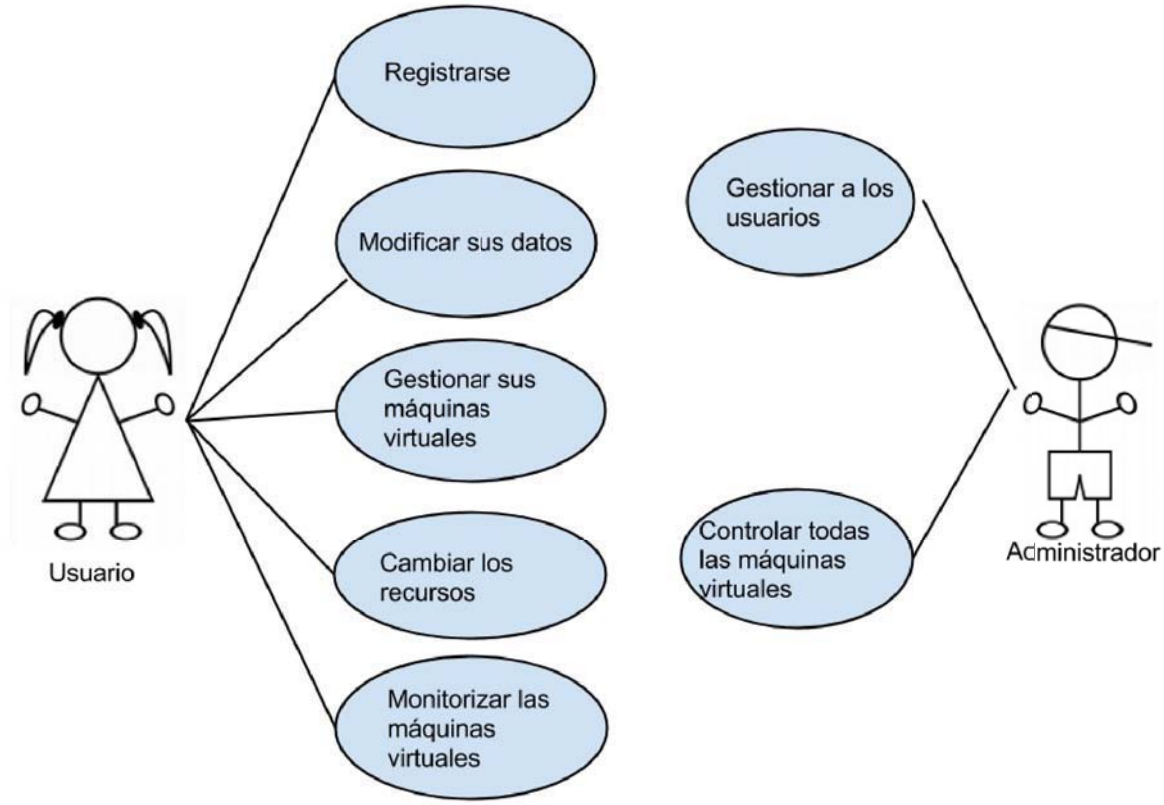


Figura 2: Diagrama de casos de uso.

Los requisitos de este proyecto se dividen en dos niveles, nivel de usuario y nivel de administrador. El administrador será el que se encargará del correcto funcionamiento de la aplicación así como de la gestión de los usuarios. En cambio, el usuario será capaz de realizar todas las opciones posibles (crear, modificar, eliminar, cambiar los recursos, acceder, etc.) sobre sus máquinas virtuales.

Registrarse	
Identificador: PCCU01	**Nombre:** Registrarse
Actor: Usuario	
Objetivo: Registrarse para poder acceder a la aplicación.	
Precondición: Estar interesado en contratar una o varias máquinas virtuales.	
Postcondición: El usuario queda registrado en la base de datos.	
Escenario básico: • El usuario rellena los datos para registrarse. • Espera recibir un correo confirmando que el registro se ha realizado satisfactoriamente.	

Tabla 1: Caso de uso Registrarse.

Modificar sus datos personales	
Identificador: PCCU02	**Nombre:** Modificar sus datos personales
Actor: Usuario	
Objetivo: Editar su información personal.	
Precondición: Querer cambiar la contraseña, el correo electrónico o algún otro dato de carácter personal.	
Postcondición: La modificación queda registrada en la base de datos.	
Escenario básico: • Se abrirá una ventana con los diferentes campos que puedes modificar. • Se rellenan los campos que quieres modificar. • Introduces la contraseña antigua, para evitar que otro individuo que no sea el propio cliente o usuario pueda modificar sus datos. • Se presiona el botón de guardar.	

Tabla 2: Caso de uso Modificar sus datos personales.

Gestionar sus máquinas virtuales	
Identificador: PCCU03	**Nombre:** Gestionar sus máquinas virtuales
Actor: Usuario	
Objetivo: Apagar, encender, reiniciar o reinstalar las máquinas virtuales	
Precondición: Estar correctamente registrado y tener alguna máquina virtual contratada.	
Postcondición: Cambia el estado de la máquina virtual.	
Escenario básico: Se abrirá una tabla con las máquinas virtuales de ese cliente, y aparecerán diferentes botones, cada uno de ellos realizara una de las siguientes funciones: • Encender la máquina virtual (Start). • Apagar la máquina virtual (Stop). • Reiniciar la máquina virtual (Reboot). • Reinstalar la máquina virtual (Reinstall).	

Tabla 3: Caso de uso Gestionar sus máquinas virtuales.

Cambiar los recursos	
Identificador: PCCU04	**Nombre:** Cambiar los recursos
Actor: Usuario	
Objetivo: Modificar el número de procesadores o el tamaño de la memoria RAM	
Precondición: Estar correctamente registrado y tener alguna máquina virtual contratada.	
Postcondición: Que los recursos de la máquina serán modificados y eso ocasionara un cambio en el rendimiento de la máquina virtual.	
Escenario básico: • Habilitar el entorno de modificar los recursos. • Aumentar o disminuir algún recurso.	

Tabla 4: Caso de uso Cambiar los recursos.

Monitorizar las máquina virtuales	
Identificador: PCCU05	**Nombre:** Monitorizar las máquinas virtuales
Actor: Usuario	
Objetivo: Controlar el rendimiento en cuanto se refiere al trafico de la red, el espacio disponible en disco, el uso de la CPU y el uso de RAM	
Precondición: Tener una máquina contratada y que este en funcionamiento.	
Postcondición: Se observaran por la pantalla las gráficas con el rendimiento de la máquina seleccionada	
Escenario básico: Acceder dentro de una máquina virtual y visualizar las diferentes gráficas que muestran la siguiente información: • Uso de la CPU. • Uso de la RAM. • Espacio disponible en el disco. • Trafico de red. Estas gráficas pueden ser de diferentes periodos de tiempo, estos periodos son los siguientes: • Actualmente (para este caso no existe gráfica). • De la última hora. • De la última semana. • Del último mes. • Del último año.	

Tabla 5: Caso de uso Monitorizar las máquinas virtuales.

Gestionar los usuarios	
Identificador: PCCU06	**Nombre:** Gestionar los usuarios
Actor: Administrador	
Objetivo: Poder eliminar del sistema usuarios que ya no utilizan el servicio.	
Precondición: Que hayan usuarios registrados.	
Postcondición: Se borra al usuario de la base de datos.	
Escenario básico: Accedes a la ventana donde te muestran todos los usuarios, con sus máquinas virtuales contratadas. Si el usuario no tiene ninguna máquina virtual contratada, se muestra la fecha en que dejo de contratar el servicio.	

Tabla 6: Caso de uso Gestionar los usuarios.

Controlar todas las máquinas virtuales	
Identificador: PCCU07	**Nombre:** Controlar todas las máquinas virtuales
Actor: Administrador	
Objetivo: Poder gestionar las máquinas de los clientes, por si ellos tienen algún tipo de dificultad o problema y no saben solucionarlo.	
Precondición: Que hayan máquinas virtuales contratadas por algún cliente.	
Postcondición: Problema de la máquina virtual resuelto.	
Escenario básico: Pantalla donde aparecen todas la máquinas virtuales que actualmente tiene que controlar el panel. Una vez filtrada la máquina virtual que le ocasiona problemas al usuario, se actúa sobre ella, solucionando el problema.	

Tabla 7: Caso de uso Controlar todas las máquinas virtuales.

Se considera que con todos los procesos anteriormente descritos se cubren todas las necesidades funcionales requeridas por parte de la aplicación.

2.2 Requisitos de datos

En este apartado se muestran los requisitos de datos del sistema. Estos abarcan toda la información que se deberá almacenar en el sistema para poder cumplir los objetivos planteados. Los datos necesarios provienen de la base de datos de la aplicación. La información puede resumirse en formularios, sobre cada uno de ellos se trabaja con la siguiente información:

Contratar una máquina virtual	
Identificador: PCRD01	**Nombre:** Contratar una máquinas virtuales
Descripción: Toda la información necesaria para crear una nueva máquina virtual.	
Dato	**Descripción**
Nombre	Nombre con el cual el cliente bautiza a la máquina.
Identificador	Número que se asigna automáticamente para identificarla.
RAM	Cantidad de memoria RAM que tiene que tener.
CPU	Número de procesadores que tiene la máquina virtual.
Disco	Capacidad total de disco duro.
Sistema Operativo	Sistema operativo que tiene que ser instalado en la máquina virtual.
Contraseña	Clave para acceder a la máquina, se genera automáticamente.
IP	Número de direcciones IP que tiene esa máquina para acceder a ella.

Tabla 8: Requisitos de datos. Contratar una máquina virtual

Registrarse	
Identificador: PCRD02	**Nombre:** Registrarse
Descripción: Toda la información necesaria para crear un nuevo usuario con acceso al panel.	
Dato	**Descripción**
Nombre	Nombre del usuario.
Usuario	Nombre con el que el usuario accede al panel.
Contraseña	Clave del usuario para acceder al panel.
Correo electrónico	Dirección de correo electrónico para informar al usuario sobre las características de las máquinas, así como para recuperar su contraseña si la ha perdido o no se acuerda de ella.

Tabla 9: Requisitos de datos. Registrarse

Con esta información que se almacena en cada formulario es suficiente para poder crear un usuario o una máquina virtual y hacer todas las funciones descritas sobre ellas.

2.4 Plataforma tecnológica

En lo referente a las necesidades técnicas los requisitos son mínimos. La aplicación debe funcionar online y sobre un navegador. Por lo que se necesita un PC capaz de conectarse a Internet pero sin necesidades de rendimiento destacables, cualquier equipo de características ofimáticas con un máximo de tres o cuatro años de antigüedad garantiza un correcto funcionamiento. Además son necesarios los siguientes periféricos: monitor, ratón y teclado.

Cabe destacar, que la infraestructura tecnológica del sistema completo requiere el uso de servidores con características muy concretas. Esta infraestructura queda fuera del alcance del proyecto.

3 Alcance

El alcance funcional del proyecto consiste en crear una aplicación web para que los usuarios sean capaces de contratar, encender, apagar, modificar y monitorizar las máquinas virtuales.

Como hemos mencionado anteriormente, el acceso a la herramienta se realiza a través de un servidor web que utiliza una base de datos en MySQL, con ciertas características especiales para adaptarse a la LOPD (Ley Orgánica de Protección de Datos de Carácter Personal). Toda esa gestión de la información queda fuera del alcance del proyecto.

Por otro lado, ha sido necesario diseñar y crear una aplicación en PHP que se encarga de gestionar los accesos a la base de datos y de las vistas que se muestran a los usuarios, así como la interacción de ellos con las vistas. Además, esta aplicación es la que utilizará la API de Proxmox para actuar sobre los servidores sobre los cuales se crearán y trabajarán las máquinas virtuales de los usuarios.

El alcance temporal del proyecto está limitado al tiempo de prácticas. En mi caso unas 300 horas, que son aproximadamente dos meses de trabajo a jornada completa.

Por otro lado, esta el alcance organizativo del proyecto, que consiste en saber que ha realizado el alumno y el supervisor de prácticas. El supervisor en este caso ha indicado que ha de realizar el alumno y como debe llevarlo a cabo.

Las restricciones del proyecto son que se tiene que utilizar el API de Proxmox para gestionar las máquinas virtuales y que el lenguaje de programación utilizado para realizar la aplicación web sea PHP, ya que se intenta utilizar herramientas de software libre. Además, los servidores sobre los cuales se instalara el motor de la aplicación son los de la empresa, pero las máquinas virtuales que creará la aplicación no tienen porque residir en esos mismos servidores, es decir, pueden ser servidores externos a la empresa. Con esto se consigue un mayor aprovechamiento de los recursos, pero se dificulta un poco la programación de la cola de tareas, ya que al no residir las maquinas en el mismo servidor que reside la aplicación, no se puede llamar a la API de Proxmox con llamadas al sistema.

4 Restricciones

Este proyecto se ha encontrado con una serie de problemas que han frenado las posibles funcionalidades de la aplicación final. Como puede observarse en los requisitos, sería posible ampliar el alcance del proyecto añadiendo nuevas características, sin embargo existían desde el principio diversas restricciones que han obligado a limitar el desarrollo final. Como consecuencia los requisitos del proyecto se han cumplido, pero no se han podido ampliar para hacer más completa la aplicación.

Este apartado enumera y describe estas restricciones.

4.1 Tiempo

Cuando se obtuvieron las especificaciones, se contaba con un tiempo limitado a tres meses para que la aplicación saliera al mercado. Así que era necesaria una primera versión funcional que cumpliera, al menos, con los requisitos mencionados. Por ese motivo algunas propuestas y funcionalidades del proyecto fueron rechazadas en beneficio de agilizar el tiempo de desarrollo lo máximo posible.

4.2 Tecnología

Como el desarrollo se encuentra integrado en una plataforma que ya se encuentra en funcionamiento, las posibles herramientas a elegir se encuentran limitadas y deben adaptarse a estas tecnologías. Además uno de los requisitos era que la aplicación web fuera desarrollada en PHP, ya que de esa forma se podría utilizar el framework CakePHP, el cual sigue el patrón de diseño **MVC** (**M**odelo **V**ista **C**ontrolador).

Otro motivo para utilizar PHP es debido a que la aplicación tiene que utilizar la API de Proxmox para gestionar las máquinas de los usuarios. Y esta API tiene soporte para algunos lenguajes de programación, entre cuales, se encuentra PHP.

Además de utilizar PHP, en el desarrollo también nos hemos visto obligados a utilizar otros lenguaje para hacer la aplicación más atractiva y mejorar el rendimiento. Estos lenguajes son los siguientes:

- JavaScript. Este lenguaje nos es útil para evitar peticiones desde la parte del cliente hacia el servidor, ya que se ejecuta en el navegador y sirve para validar los datos sin tener que enviarlos. Además, mediante el JavaScript conseguimos que las páginas se han dinámicas.
- JQuery. Ya que se trata de un lenguaje flexible y rápido para el desarrollo web. También tiene una excelente integración con el AJAX y esto nos permite recargar partes de la interfaz sin necesidad de recargar toda la pagina. Por ejemplo, ir actualizando los datos de las gráficas sobre as máquinas virtuales sin tener que ir refrescando la página para ver la utilización de sus recursos hardware.
- HTML5. Este lenguaje es el que se utiliza para realizar las interfaces de la aplicación.
- CSS y Bootstrap. Se utiliza para dar estilo al HTML y hacer las interfaces más atractivas, intuitivas y fáciles de utilizar para los usuarios de la aplicación.

En cuanto a la base de datos, la hemos implementado en MySQL ya que es un software gratuito, proporciona integridad referencial y la relación entre calidad, usabilidad y precio es muy buena. Además se integra perfectamente en nuestro proyecto como se explicara en apartados posteriores.

Por otro lado, hemos utilizado el servidor Apache, ya que entre otras ventajas, permite implementar conexiones seguras y es uno de los más utilizados y extendidos.

Para la gestión de las máquinas virtuales, hemos empleando Proxmox, que se trata de una plataforma de virtualización para máquinas virtuales. Nosotros, a través de ella, gestionamos las máquinas virtuales de la aplicación web.

Finalmente, el entorno de desarrollo que hemos utilizado es PhpStrom ya que es un editor para los lenguajes mencionados anteriormente, y además, es compatible con varias versiones de PHP.

4.3 Experiencia

La experiencia del estudiante está centrada en los conocimientos adquiridos durante el tiempo que se ha cursado el Grado en Ingeniería Informática. En el transcurso del grado se han adquirido competencias a través de breves experiencias con Java, Gestión de Proyectos o Bases de Datos. Sin embargo, el framework CakePHP y la programación en PHP para abordar el proyecto es algo para lo que es necesario un periodo más amplio de aprendizaje o experiencia. Por ese motivo, uno de los factores restrictivos relacionados con los dos anteriores, es la falta de experiencia del estudiante ya que para el aprendizaje de la herramienta, ha sido necesario que el estudiante invierta un tiempo superior al que podría esperarse para proyectos con requisitos similares.

4.4 Presupuesto

Asumiendo que el salario medio de un programador junior es de 1300€ aproximadamente, la duración de este proyecto son unas 300 horas (Dos meses aproximadamente a jornada completa), esto haría un total de 2600€. El coste para la empresa sería de 3380€, un 30% más debido a los costes laborales que tiene que pagar la empresa por el contrato.

Por otro lado, se encuentra el coste de amortización del hardware, que se calcula mediante la siguiente formula:

((Valor de compra – Valor residual) / horas laborales durante 4 años) x Número de horas dedicadas al proyecto

Sustituyendo las variables de la formula anterior, obtenemos:

((500 - 0) / 2120) x 300 = ***70,75 €***

Donde:

- 500 son los euros que cuesta un PC.
- 0 es el valor que tendrá el PC a los cuatro años de su compra.
- 2120 son las horas laborales de cuatro años suponiendo que el año tiene de 53 semanas.
- 300 son las horas de dedicación al proyecto.

Se ha podido observar que el hardware que se ha contabilizado es referente al equipo de trabajo para desarrollar la aplicación. En cuanto al servidor, como hemos mencionado en apartados anteriores se pretende aprovechar los servidores de la empresa. Estos servidores se supone que ya están amortizados y, por lo tanto, el coste que se imputa en el proyecto se puede considerar nulo, es decir, cero euros. En cuanto a los servidores que harán falta para alojar las máquinas virtuales, en principio se pretende alquilarlos por uso. Ya que comprarlos supone un gasto muy elevado y

todavía no se sabe con certeza el existo que tendrá la aplicación entre los usuarios de internet.

En cuanto al coste de las instalaciones y del software empleado, el coste sería prácticamente nulo, ya que el software utilizado para desarrollar la aplicación es libre. En resumen, el coste referente al software y a la instalación de la infraestructura es de cero euros.

Recurso	Gasto
Hardware	70,75€
Software	0€
Personal	3380€
Total:	**3450,75€**

Tabla 10: Presupuesto detallado del proyecto.

5 Planificación

Una vez se han definido correctamente los objetivos, requisitos, alcance y restricciones del desarrollo implicado en el proyecto se va a mostrar la planificación realizada para llevarlo a cabo. Para ello se va a comenzar presentando el diagrama EDT (Estructura del Desglose del Trabajo), seguidamente se dará una descripción de las tareas incluidas y, finalmente, se presentará un diagrama de Gantt donde se establecen fechas concretas para los diferentes hitos.

5.1 EDT

El diagrama EDT presenta la descomposición jerárquica de todo el trabajo del proyecto en componentes más pequeños. Comienza presentándose como un listado de las tareas a realizar y, a continuación, se representa el diagrama en formato gráfico:

Desarrollo del proyecto:

1 Planificación
- 1.1 Definir objetivos.
- 1.2 Delimitar el alcance.
- 1.3 Valorar las restricciones.
- 1.4 Generar la planificación.

2 Análisis
- 2.1 Definición de casos de uso.
- 2.2 Desarrollo de los requisitos.

3 Diseño
- 3.1 Diagrama de actividades.
- 3.2 Diagrama de clases.
- 3.3 Especificación de la interfaz.

4 Implementación
- 4.1 Desarrollo de la base de datos
- 4.2 Desarrollo de la interfaz de usuario

5 Pruebas
- 5.1 Pruebas de expertos.
- 5.2 Pruebas de usuarios finales.

6 Instalación
- 6.1 Integración de la aplicación en la plataforma.

7 Mantenimiento

La siguiente figura representa gráficamente el diagrama EDT (Figura 3):

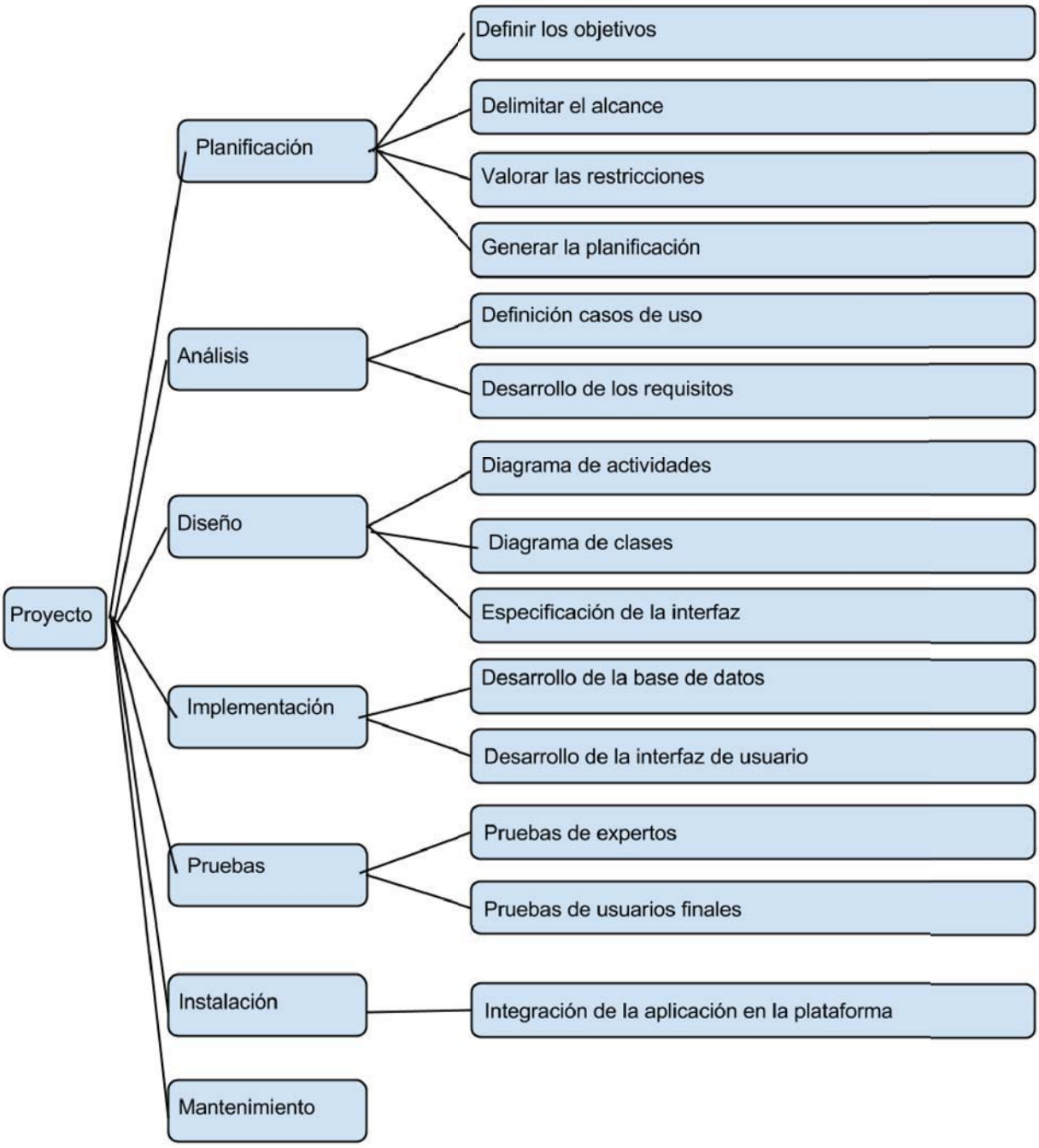


Figura 3: Diagrama EDT.

A continuación se describen las actividades finales extraídas del diagrama EDT.

- **Definir objetivos**. Llevar a cabo una serie de reuniones con el departamento de I+D de la empresa.
- **Delimitar el alcance**. Ya que el proyecto está limitado en cuanto al número de horas de dedicación, es importante reunirse para delimitar las funcionalidades que son más importantes para poder lanzar el producto al mercado aunque esté no tenga todas las funcionalidades deseadas.
- **Valorar las restricciones**. Una vez están definidos los objetivos y el alcance pueden

valorarse las restricciones, analizando además los riesgos que pueden presentarse en el proceso de desarrollo.

- **Generar la planificación**. Para finalizar esta fase de planificación es necesario generar los documentos donde se definen las tareas a llevar a cabo y los tiempos estimados. En este punto debe generarse el documento donde se refleja la planificación inicial.
- **Definición de casos de uso.** Ya en la fase de análisis se estudian los posibles actores y casos de uso que van a intervenir.
- **Desarrollo de los requisitos**. Con los casos de uso generados se generan los requisitos. Para ello es necesario reunirse y estudiar otras aplicaciones similares a la que estamos desarrollando. En este punto debe generarse el documento referente a la especificación de requisitos, éste servirá de referencia durante todo el tiempo de desarrollo.
- **Diagrama de actividades.** Ya en la fase de diseño es necesario generar los diagramas de actividades que facilitan la posterior implementación.
- **Diagramas de clases.** Del mismo modo es necesario crear un diagrama de clases que permita describir la estructura del sistema.
- **Especificación de la interfaz**. Haciendo uso de prototipos se realizan bocetos conceptuales de cómo se estructurará visualmente la información en la pantalla. Para ello es necesario mantener contacto con el departamento de diseño para terminar hacendó una interfaz atractiva.
- **Desarrollo de la base de datos**. Ya en la fase de implementación se desarrolla la base de datos tal y como se ha definido en la fases anteriores.
- **Desarrollo de la interfaz de usuario.** Igual que en el caso anterior, en la fase de implementación se desarrolla visualmente la interfaz tal y como ha sido definida en la fase anterior.
- **Pruebas de expertos.** Aunque van a realizarse pruebas durante todo el proceso de implementación, en esta fase se integran las pruebas de los expertos para detectar posibles errores de usabilidad o mejoras en la aplicación.
- **Pruebas de usuarios finales.** Con una versión de la herramienta funcional y validada por los expertos, se someterá a pruebas mediante el uso, con un grupo de usuarios comunes que pueden realizar pruebas y detectar errores o aspectos susceptibles de mejora que hayan pasado desapercibidos en las fases anteriores.
- **Integración de la aplicación en la plataforma.** Una vez que todas las fases anteriores hayan sido concluidas con éxito, se intenta integrar la aplicación en la plataforma para la cual fue creada.

Con esto quedan definidas todas las tareas a llevar a cabo en el desarrollo.

5.2 Gantt

A continuación se presenta un diagrama de Gantt en el que se especifican las tareas presentadas en el EDT con tiempos y dependencias asociadas.

En primer lugar se presenta la tabla donde se muestran las fechas y las dependencias en formato de tabla (Figura 4):

Núm...	ID	Nombre	Antecesores	Fecha de inicio	Fecha de fin	Horas
1	0	Poyecto final de grado		7/03/14	11/07/14	423
1.1	2	Desarrollo de la propuesta técnica		7/03/14	21/03/14	47
1.1.1	3	Inicio		7/03/14	11/03/14	11
1.1.1.1	7	Definir el proyecto con el tutor y el supervisor		7/03/14	7/03/14	1
1.1.1.2	8	Definir el método de trabajo y documentación	7	10/03/14	10/03/14	5
1.1.1.3	9	Definir el formato y los estándares de trabajo	8	11/03/14	11/03/14	5
1.1.2	4	Documentación del proyecto		12/03/14	17/03/14	16
1.1.2.1	10	Revisar contexto y buscar información	9	12/03/14	14/03/14	15
1.1.2.2	11	Identificar alcance y objetivos	10	17/03/14	17/03/14	5
1.1.3	5	Planificación del proyecto		18/03/14	21/03/14	20
1.1.3.1	12	Definir tareas y estimar fechas	11	18/03/14	18/03/14	5
1.1.3.2	13	Crear diagrama de Gantt	12	19/03/14	19/03/14	5
1.1.3.3	14	Documentar la propuesta del proyecto	13	20/03/14	20/03/14	5
1.1.3.4	15	Entregar la propuesta técnica	14	21/03/14	21/03/14	5
1.2	6	Desarrollo técnico del proyecto		24/03/14	9/06/14	275
1.2.1	17	Definir requisitos del proyecto		24/03/14	25/03/14	10
1.2.1.1	18	Analisis del mercado	15	24/03/14	24/03/14	5
1.2.1.2	19	Definir requisitos tecnológicos	18	25/03/14	25/03/14	5
1.2.2	20	Análisis y diseño		25/03/14	3/04/14	35
1.2.2.1	24	Identificar a los usuarios	19	26/03/14	26/03/14	5
1.2.2.2	25	Diseño de las interfaces de usuario	24	27/03/14	27/03/14	5
1.2.2.3	26	Crear diagramas de clases	18	25/03/14	31/03/14	10
1.2.2.4	27	Documentar clases	18	25/03/14	2/04/14	10
1.2.2.5	28	Validación del análisis y el diseño	25,26,27	3/04/14	3/04/14	5
1.2.3	21	Desarrollo		4/04/14	6/06/14	225
1.2.3.1	29	Programación de la aplicación	28	4/04/14	2/06/14	210
1.2.3.2	30	Pruebas	29	3/06/14	6/06/14	15
1.2.4	22	Puesta en marcha		9/06/14	9/06/14	5
1.2.4.1	23	Entrega final	30	9/06/14	9/06/14	5
1.3	16	Documentación y presentación del TFG		10/03/14	11/07/14	101
1.3.1	31	Redacción de informes quincenales		10/03/14	9/06/14	40
1.3.2	32	Redacción de memoria técnica		10/03/14	30/06/14	45
1.3.3	33	Entrega de la memoria técnica	32	1/07/14	1/07/14	0
1.3.4	34	Preparación de la presentación oral	33	2/07/14	10/07/14	15
1.3.5	35	Presentación oral	34	11/07/14	11/07/14	1

Figura 4: Diagrama de Gantt, tabla de tareas.

A continuación, se presenta el diagrama en formato de barras (Figura 5) para facilitar la visualización de todo el proceso de desarrollo del proyecto. Se han incluido el nombre de las tareas y las fechas.

Núm.	ID	Nombre	Antecesores	Fecha de inicio	Fecha de fin	Horas
1	0	Proyecto final de grado		7/03/14	11/07/14	423
1.1	2	Desarrollo de la propuesta técnica		7/03/14	21/03/14	47
1.1.1	3	Inicio		7/03/14	11/03/14	11
1.1.1.1	7	Definir el proyecto con el tutor y el supervisor		7/03/14	7/03/14	1
1.1.1.2	8	Definir el método de trabajo y documentación	7	10/03/14	10/03/14	5
1.1.1.3	9	Definir el formato y los estándares de trabajo	8	11/03/14	11/03/14	5
1.1.2	4	Documentación del proyecto		12/03/14	17/03/14	16
1.1.2.1	10	Revisar contexto y buscar información	9	12/03/14	14/03/14	15
1.1.2.2	11	Identificar alcance y objetivos	10	17/03/14	17/03/14	5
1.1.3	5	Planificación del proyecto		18/03/14	21/03/14	20
1.1.3.1	12	Definir tareas y estimar fechas	11	18/03/14	18/03/14	5
1.1.3.2	13	Crear diagrama de Gantt	12	19/03/14	19/03/14	5
1.1.3.3	14	Documentar la propuesta del proyecto	13	20/03/14	20/03/14	5
1.1.3.4	15	Entregar la propuesta técnica	14	21/03/14	21/03/14	5
1.2	6	Desarrollo técnico del proyecto		24/03/14	9/06/14	275
1.2.1	17	Definir requisitos del proyecto		24/03/14	25/03/14	10
1.2.1.1	18	Análisis del mercado	15	24/03/14	24/03/14	5
1.2.1.2	19	Definir requisitos tecnológicos	18	25/03/14	25/03/14	5
1.2.2	20	Análisis y diseño		25/03/14	3/04/14	35
1.2.2.1	24	Identificar roles usuarios	19	26/03/14	26/03/14	5
1.2.2.2	25	Diseño de las interfaces de usuario	24	27/03/14	27/03/14	5
1.2.2.3	26	Crear diagramas de clases	18	25/03/14	31/03/14	10
1.2.2.4	27	Documentar clases	18	25/03/14	26/04/14	10
1.2.2.5	28	Validación del análisis y el diseño	25,26,27	3/04/14	3/04/14	5
1.2.3	21	Desarrollo		4/04/14	6/06/14	225
1.2.3.1	29	Programación de la aplicación	28	4/04/14	28/05/14	210
1.2.3.2	30	Pruebas	29	30/05/14	6/06/14	15
1.2.4	22	Puesta en marcha		9/06/14	9/06/14	5
1.2.4.1	23	Entrega final	30	9/06/14	9/06/14	5
1.3	16	Documentación y presentación del TFG		10/03/14	11/07/14	101
1.3.1	31	Redacción de informes quincenales		10/03/14	9/06/14	40
1.3.2	32	Redacción de memoria técnica		10/03/14	30/06/14	45
1.3.3	33	Entrega de la memoria técnica	32	1/07/14	1/07/14	0
1.3.4	34	Preparación de la presentación oral	33	2/07/14	10/07/14	15
1.3.5	35	Presentación oral	34	11/07/14	11/07/14	1

Figura 5: Diagrama de Gantt.

6 Diseño

En este capítulo se presenta todo lo que envuelve al diseño de la aplicación. En primer lugar se muestra el diseño de la interfaz gráfica. Y, seguidamente, se muestra la parte de este referente al diseño del sistema.

6.1 Diseño de la interfaz

Prácticamente, todas las interfaces realizadas en el proyecto están dedicadas a que las manejen los usuarios, por lo tanto, se ha intentando conseguir que las interfaces sean muy intuitivas, fáciles de usar y de entender.

Hay algunas interfaces, como la de la figura 6, que los usuarios no tienen acceso a ellas. Estas interfaces son las que pertenecen para el administrador, ya que como su nombre indica, se encarga de administrar la aplicación y por lo tanto, tiene que ser capaz de realizar más acciones que las de los usuarios. Estas acciones son las que se han explicado en los casos de uso (administrar usuarios, controlar todas las máquinas de la aplicación, etc.).

A continuación se muestra la interfaz que sirve para gestionar los usuarios de la aplicación. Solo el administrador tiene permisos para poder acceder y realizar funciones sobre el resto de usuarios de la aplicación.

Figura 6: Gestión de los usuarios registrados en la aplicación.

Antes de profundizar más en el diseño de las diferentes vistas que tendrá la aplicación, se presenta una visión general de esta, en la cual se sigue el flujo que tendrá la aplicación entre las diferentes vistas, en este caso utilizaremos el nombre de la interfaz y el número de figura de los prototipos que se explicarán en las siguientes secciones.

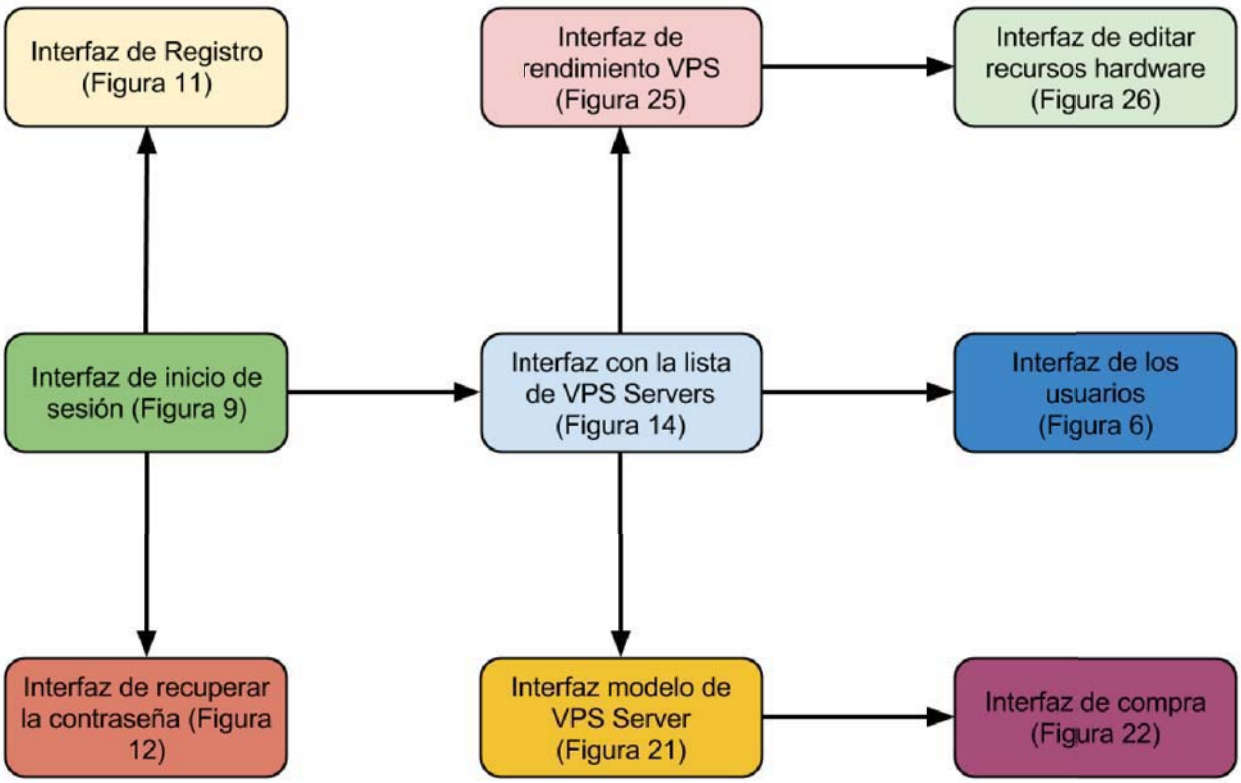


Figura 7: Visión general del flujo de la aplicación.

6.1.1 Elementos comunes

La principal característica que se irá repitiendo a lo largo de las diferentes interfaces de la aplicación son los colores. La elección de los colores fue una tarea rápida. Por parte de la empresa se recomendó el uso de un tema de bootstrap (Ace - Responsive Admin Template)[4], ya que de esa forma no tenía que emplear tanto tiempo en el diseño de las interfaces y podía dotar de más funcionalidades la aplicación.

Después de elegir el tema, los elementos que se necesitaban añadir (gráficas, sliders, etc.) tienen que integrarse con el tema elegido, y por lo tanto, se tienen que elegir colores que se distingan y apariencias similares a los otros elementos. Con esto se pretende conseguir que las personas que sufren daltonismo sean capaz de navegar sin problemas por la aplicación.

A continuación se observa una vista del tema que se utilizó para hacer la aplicación web:

Figura 8: Tema de Bootstrap. Ace

6.1.2 Vistas Principales

Las vistas principales serán las que aparecerán al iniciar la aplicación y desde las que se navegará hacia las principales funciones de esta.

Cuando se accede a la aplicación se mostrará una vista, a la que he llamado "Inicio de sesión", desde la cual una vez nos hemos registrado con éxito se podrá acceder al resto de interfaces.

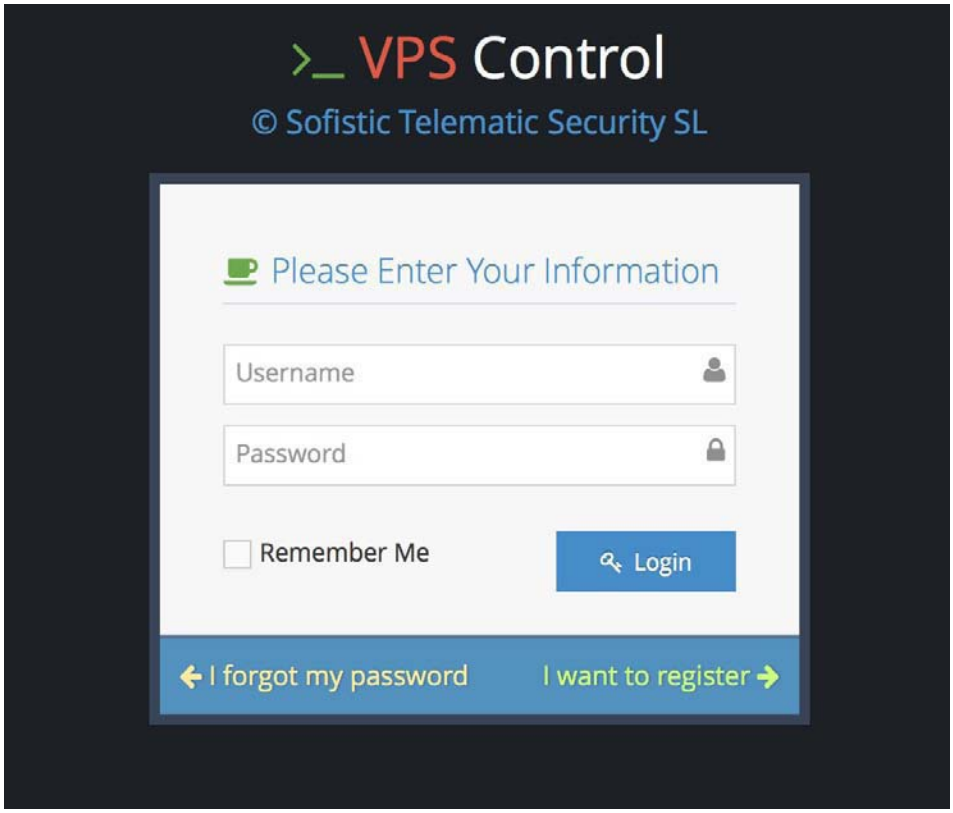


Figura 9: Interfaz de inicio de sesión.

Si nos equivocamos a la hora de introducir tu usuario o la contraseña, aparecerá un mensaje de error informando que la contraseña o el usuario no son correctos. Este mensaje sería el siguiente:

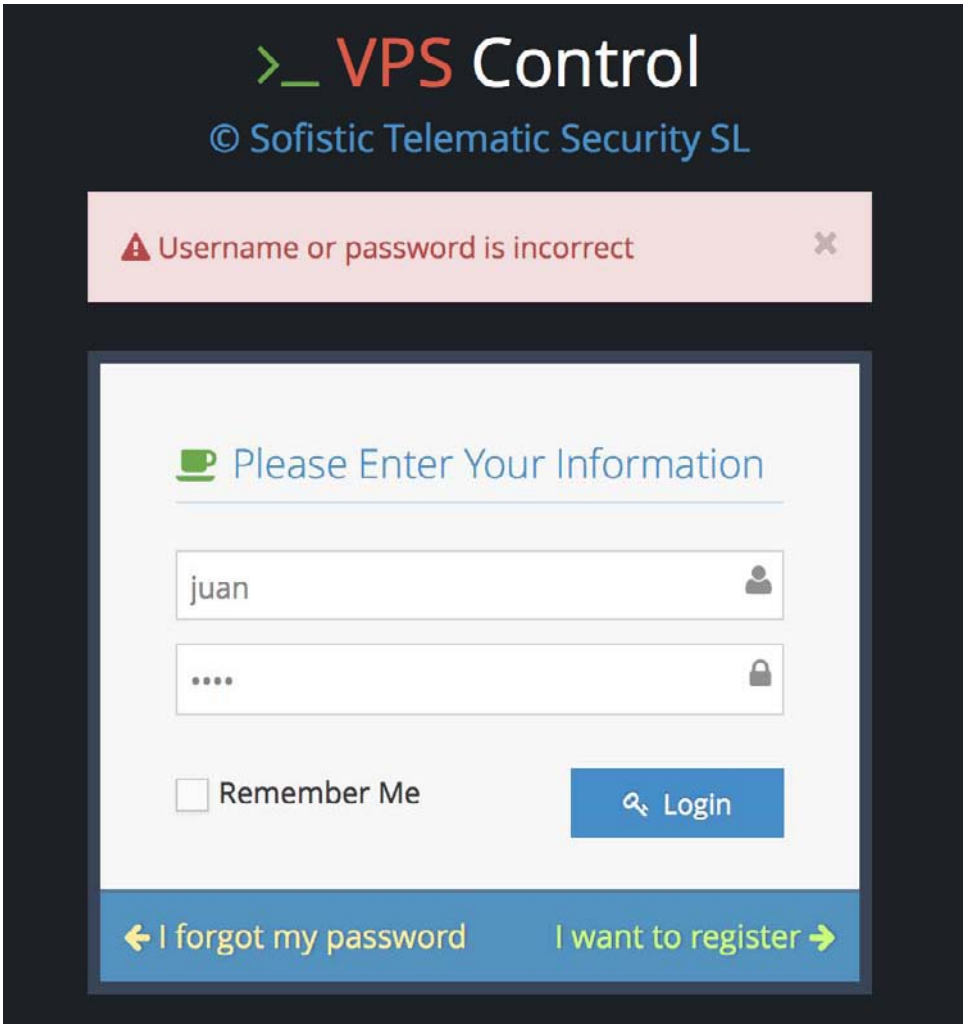


Figura 10: Mensaje de error al iniciar sesión.

Si es la primera vez que se accede a la aplicación, no habrá ningún usuario creado y por lo tanto el primer paso será registrarse. Para ello solo habrá que presionar el botón "I want to register" y aparecerá la interfaz de la figura 11:

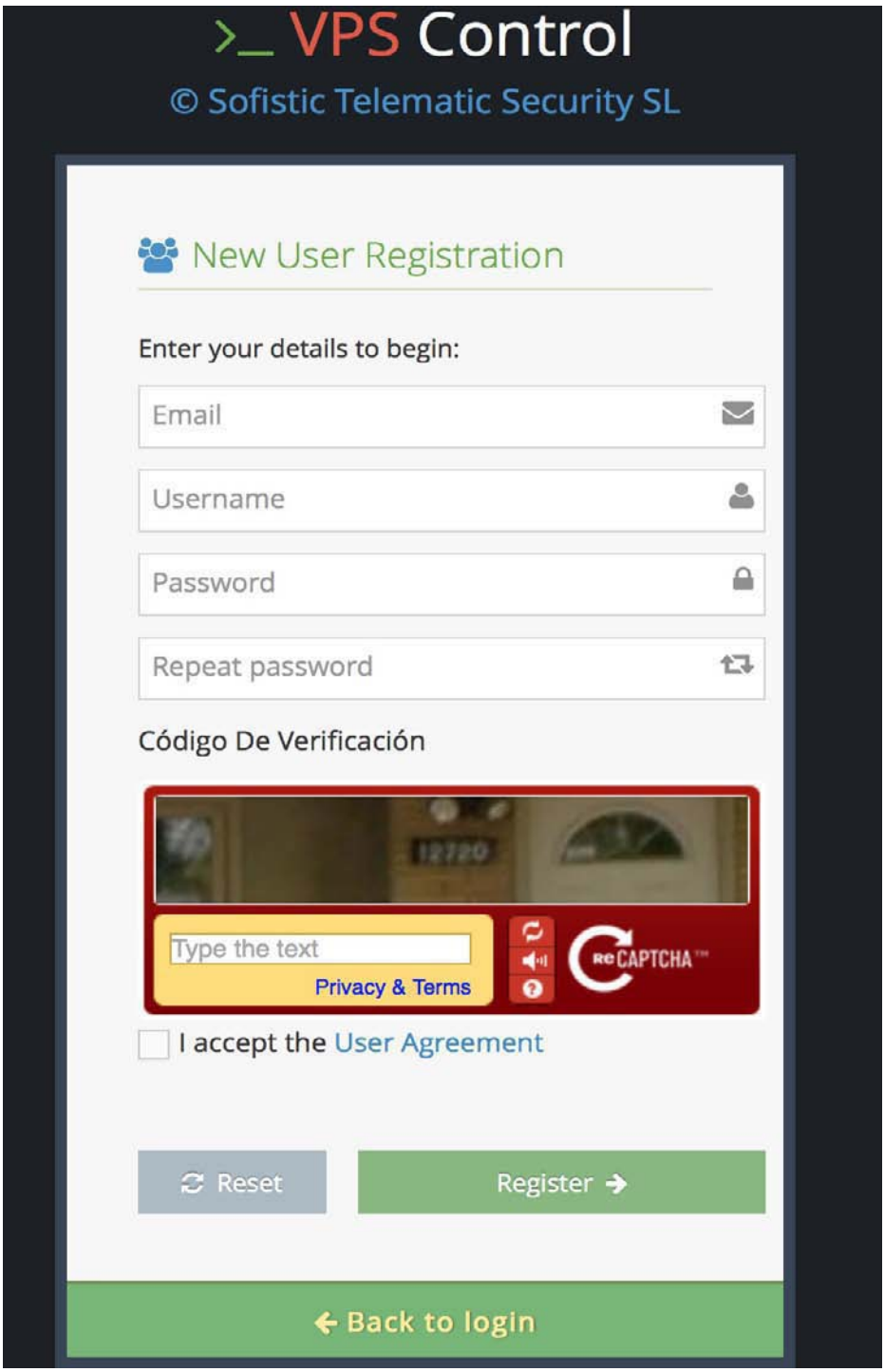


Figura 11: Interfaz de Registro.

El formulario que aparece en la interfaz anterior (Figura 11), también tiene control de errores para que los usuarios no introduzcan datos incorrectos. El control de errores aquí se basa en mirar que los datos introducidos sean correctos, por ejemplo, en el campo del correo electrónico, se asegura que se introduce una dirección de correo válida, ya que posteriormente se recibirá un correo para confirmar la cuenta en VPS Control. Por otro lado, también nos aseguramos que el usuario acepta las condiciones de la página. Y finalmente, se ha introducido un captcha para asegurarnos de que el usuario es una persona física.

Otra interfaz importante, es la de recuperar la contraseña en caso de que el usuario no se acuerde de ella. Esta interfaz, al igual que la anterior, también valida los datos y tiene un captcha. Esta interfaz es la que se encuentra en la figura 12.

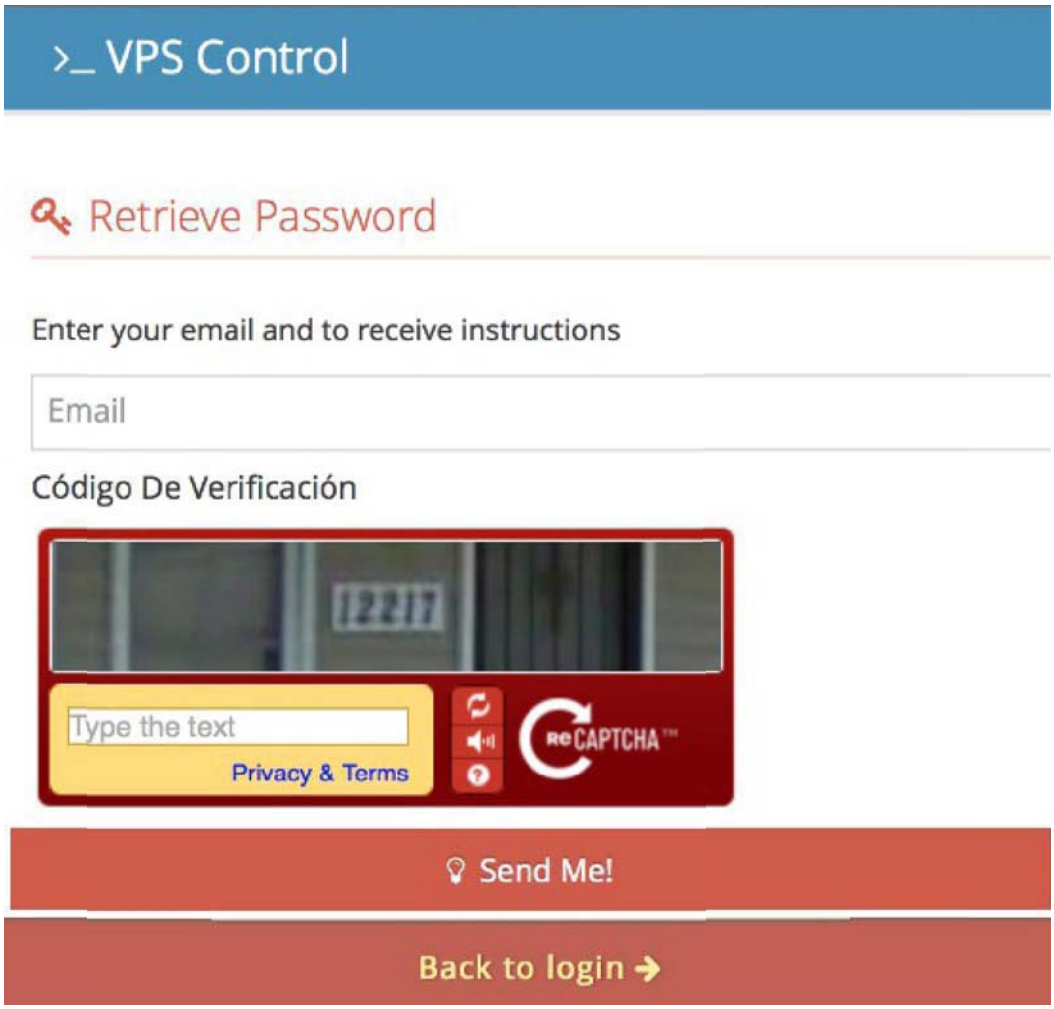


Figura 12: Interfaz de Recuperar Contraseña.

Como se observa en la interfaz, para recuperar la contraseña solo hace falta introducir el correo electrónico ya que se generara un token y esté será enviado mediante un correo electrónico a la persona a la que pertenezca ese correo. Entonces, una vez reciba el correo, se hace click en el enlace y se cargará una pantalla para que introduzca la nueva contraseña.

Si el correo electrónico no pertenece a ningún usuario de la aplicación, no se enviará el correo a la dirección introducida.

Un ejemplo del correo que se le envía al usuario para recuperar la contraseña sería:

Figura 13: Correo de recuperación de la contraseña.

Una vez nos hemos registrado en la aplicación, la interfaz que se observa será la siguiente:

Figura 14: Interfaz con el listado de VPS Server de un usuario.

En este caso se observa que el usuario tiene tres máquinas virtuales vps creadas. De las cuales dos de ellas están en funcionamiento y la otra en este momento se encuentra parada.

Figura 15: Estados de las máquinas virtuales.

Además se puede ver que también informa de que sistema operativo tiene instalado cada máquina y cual es su ip para poder acceder a ella mediante ssh. En este caso, las máquinas tiene dos ip, pero como veremos posteriormente, a una máquina virtual le podemos asignar todas las ip que queramos.

Otro aspecto importante de esta tabla es que informa de cuando se termina el contrato de cada máquina para que se vuelva a contratar si le interesa al usuario.

Y finalmente, se observan una serie de botones que ayudan a gestionar cada máquina de forma precisa. La misión de los botones son las siguientes:

- Arrancar la máquina virtual.

Figura 16: Botón de arrancar.

- Parar la máquina virtual.

Figura 17: Botón de parar.

- Reiniciar la máquina virtual.

Figura 18: Botón de reiniciar.

- Reinstalar la máquina virtual.

Figura 19: Botón de reinstalar.

- Opciones de la máquina virtual.

Figura 20: Botón de opciones.

Por último, en la parte superior de la tabla, al lado derecho, se aprecia un botón (link) que si se presiona te redirige a otra interfaz en la cual se empieza a crear una máquina virtual. Esta interfaz es la de la figura 21.

Figura 21: Interfaz para elegir hardware y características principales de la máquina virtual.

En esta interfaz podemos elegir entre cuatro modelos distintos de máquinas virtuales. Como se puede observar, las diferencias se encuentran en el hardware dedicado a cada maquina, ya que cada modelo superior tiene más número de procesadores, más megabytes de memoria RAM o más espacio de disco duro. Los usuarios, para contratar una máquina, tienen que elegir el modelo que más se les acople a sus necesidades, ya que dependiendo del modelo, el precio cambia. Además, como después observaremos en otra interfaz, el número de procesadores y el tamaño de la memoria RAM se pueden aumentar o disminuir sin ninguna complicación.

De esta interfaz no hay nada más que explicar, una vez el usuario tenga claro que pack es el que necesita, presiona el botón de comprar (Buy) y entonces será redirigido a la interfaz de la figura 22:

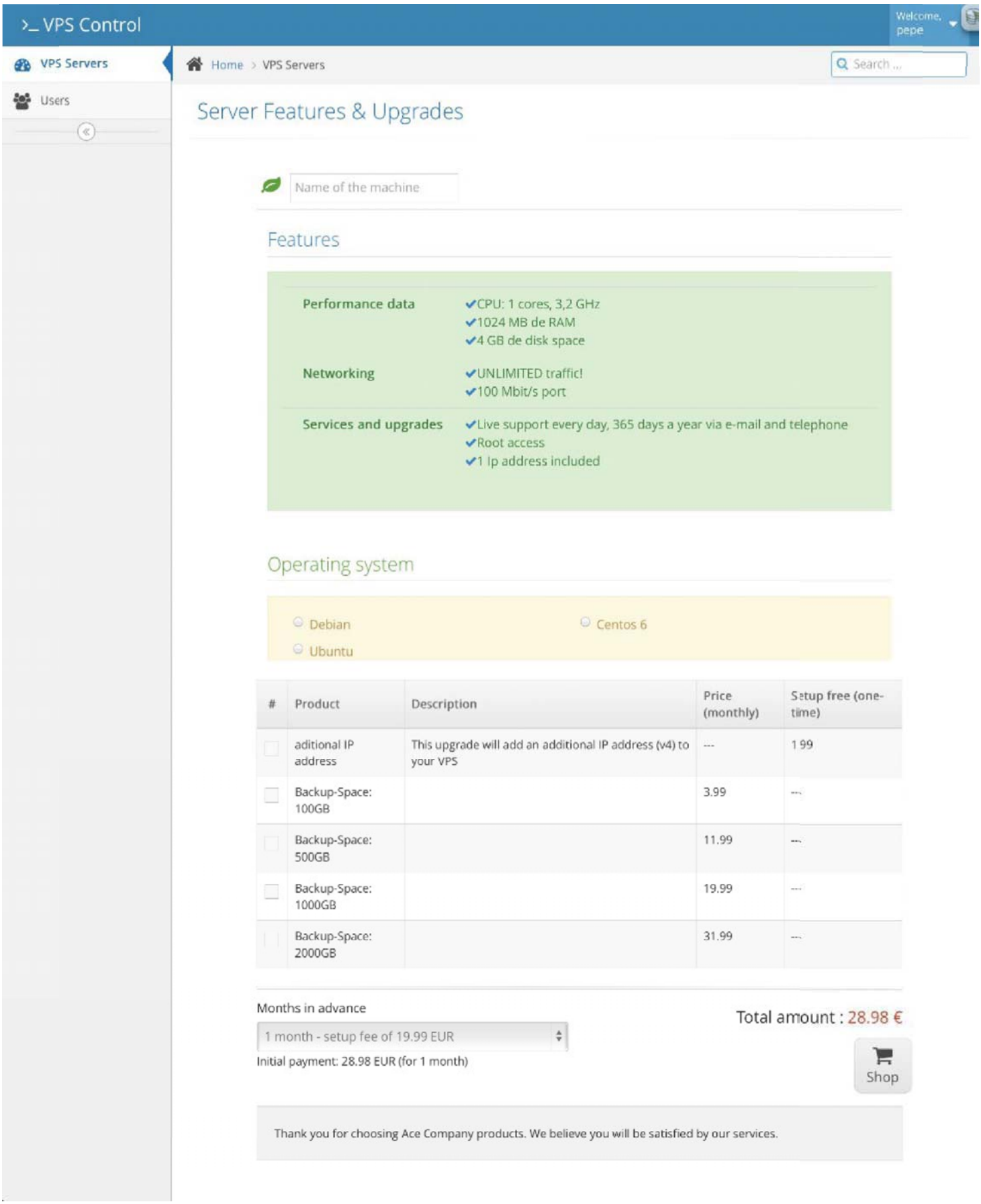


Figura 22: Interfaz de compra. Opciones adicionales sobre las máquinas virtuales vps.

Esta pantalla nos muestra las opciones que tenemos disponibles para la máquina que se ha contratado en la pantalla anterior.

El primer elemento de la pantalla que nos encontramos en el formulario es un campo de texto. Este campo de texto sirve para que el usuario introduzca el nombre con el que quiere bautizar a la máquina virtual. Este campo es obligatorio.

La siguiente información que nos encontramos, son las características del pack que hemos elegido anteriormente.

A continuación, encontramos otro campo de entrada que es para elegir el sistema operativo que queremos que se instale en la máquina que acabamos de contratar. En este caso, solo hay tres para elegir, pero eso es debido a que es una versión de pruebas y con estos tres sistemas era suficiente para probar que funcionaba correctamente. Este campo como sucede con el anterior, también es obligatorio.

Luego encontramos una tabla, donde seleccionamos características extra para nuestra máquina. Estas características son:

- Obtener mas de una IP para la maquina Virtual.
- Contratar un espacio para copias de seguridad de:
    - 100 Gigabytes.
    - 500 Gigabytes.
    - 1000 Gigabytes.
    - 2000 Gigabytes.

Como se observa, por contratar algunas de las siguientes opciones, se tiene que abonar una cantidad extra de dinero, ya sea mensualmente o anualmente.

Y finalmente, el último campo a elegir es el período de tiempo en el cual queremos disponer de esa máquina. Además, a mayor período de tiempo, menores son los gastos que la empresa cobrará por el mantenimiento de los servidores donde están alojadas las máquinas virtuales.

Al final de la interfaz, en el lado derecho, se observa el total que se tiene que pagar por la máquina con las características y condiciones elegidas. Si se está conforme solo hay que aceptar el botón de comprar (Shop) y la máquina se creará automáticamente.

Para crear la máquina, antes se tiene que realizar el pago del importe que se muestra por pantalla. Si el pago es correcto, la máquina se creará. En caso contrario informará del error y volvemos a la pantalla para que se vuelva a introducir los datos.

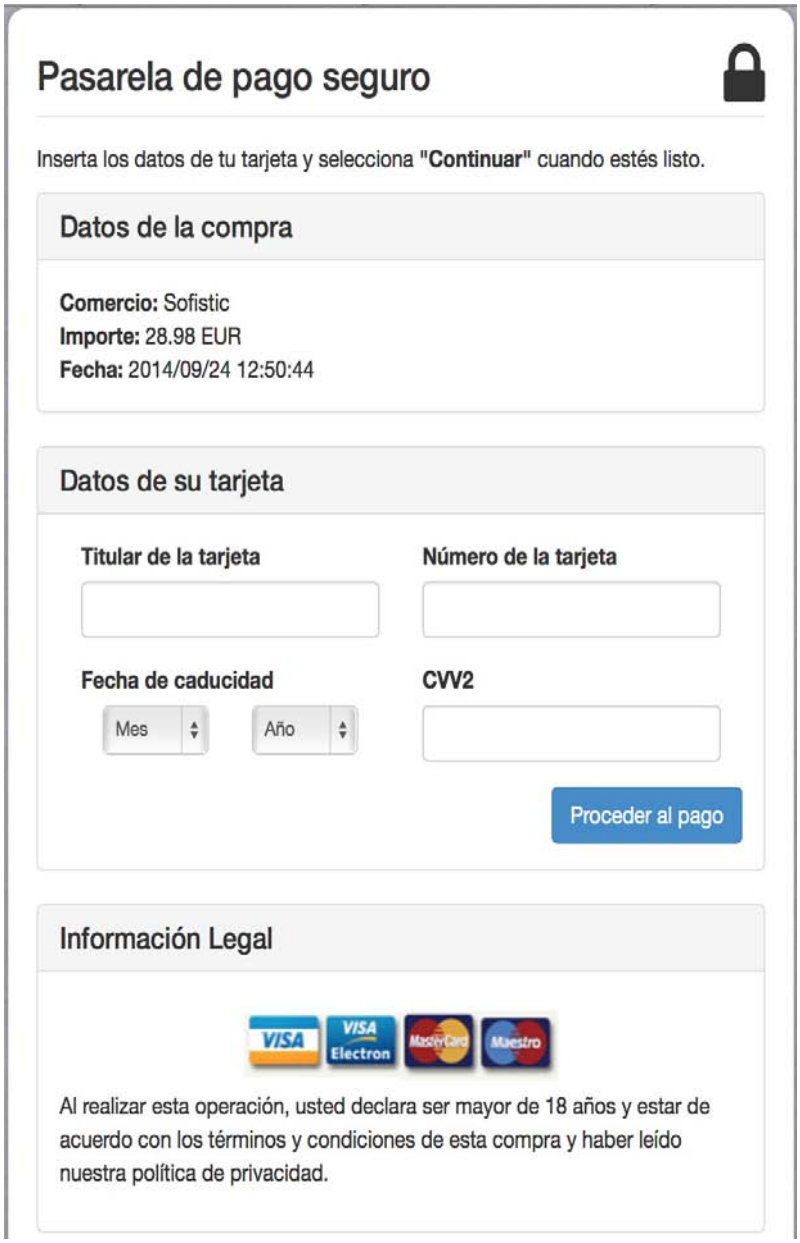
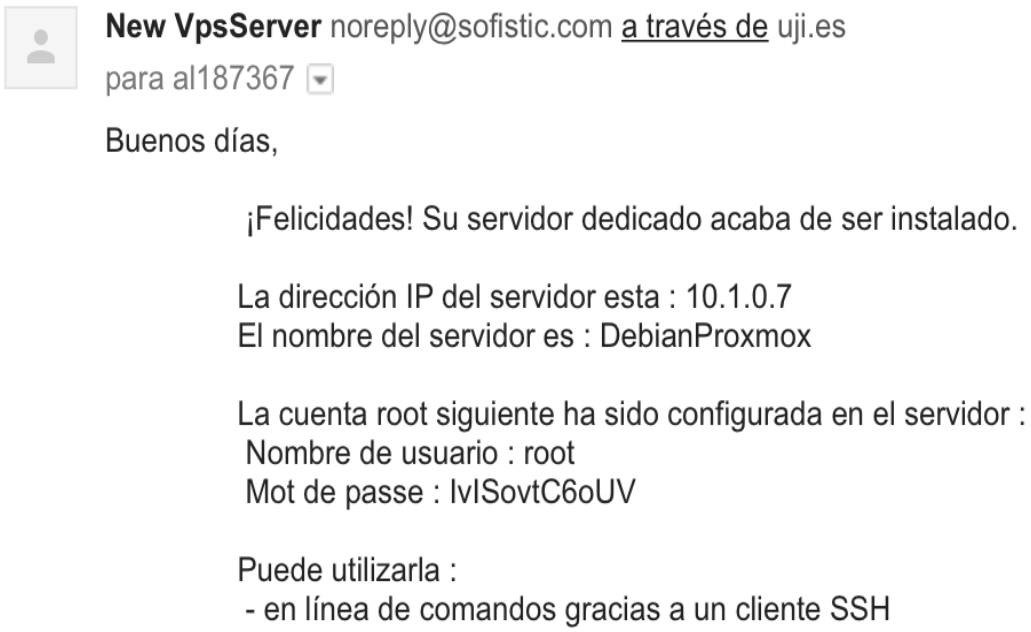


Figura 23: Interfaz de la pasarela segura de pago.

Una vez que el cobro se ha efectuado correctamente, el usuario recibirá un correo electrónico con el nombre de la máquina, la dirección o direcciones IPs que le han sido concedidas, el nombre de usuario y una contraseña para acceder a ella mediante ssh[6].

El formato del correo recibido por el usuario es el siguiente:



New VpsServer noreply@sofistic.com a través de uji.es
para al187367

Buenos días,

¡Felicidades! Su servidor dedicado acaba de ser instalado.

La dirección IP del servidor esta : 10.1.0.7
El nombre del servidor es : DebianProxmox

La cuenta root siguiente ha sido configurada en el servidor :
Nombre de usuario : root
Mot de passe : IvISovtC6oUV

Puede utilizarla :
- en línea de comandos gracias a un cliente SSH

Figura 24: Ejemplo de correo de confirmación de la creación de una máquina virtual vps.

La máquina que se ha creado se añadirá a la tabla donde están todas las máquinas del usuario listadas. Si ahora pretendemos entrar a ver el estado de la máquina o cambiar alguna característica hardware, solo tenemos que presionar el botón de opciones (Figura 20, vista anteriormente).

Una vez presionado el botón, esté nos enviará a la pantalla de la figura 26:

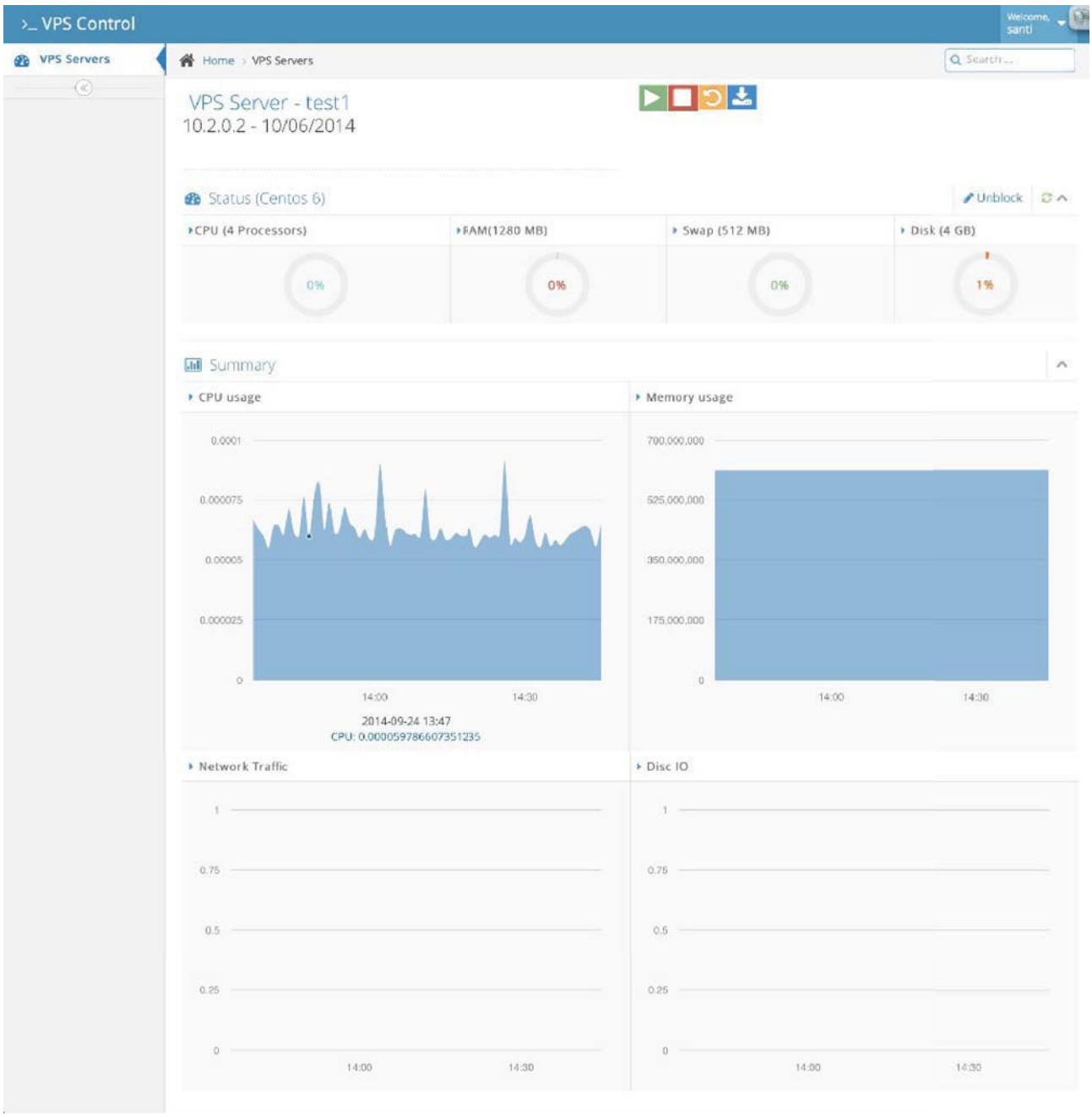


Figura 25: Interfaz del rendimiento de una máquina virtual vps.

En esta interfaz nos encontramos con los mismos botones vistos en las figuras 16, 17, 18 y 19. Al lado de estos elementos, volvemos a encontrar el nombre de la máquina, así como su dirección IP.

En la figura siguiente se encuentra representado el rendimiento o uso de los elementos hardware de la máquina. Esta medición se hace en tiempo real y se va actualizando cada 10 segundos mediante la tecnología AJAX.

Un elemento importante de esta interfaz es el que se encuentra arriba del todo y en la parte derecha. Se trata de un botón denominado “Unlock” (Figura 27), que sirve para habilitar la edición de los componentes hardware (Número de procesadores y RAM).

Cuando este botón se presiona, la interfaz que acabamos de explicar pasa a tener el siguiente formato:

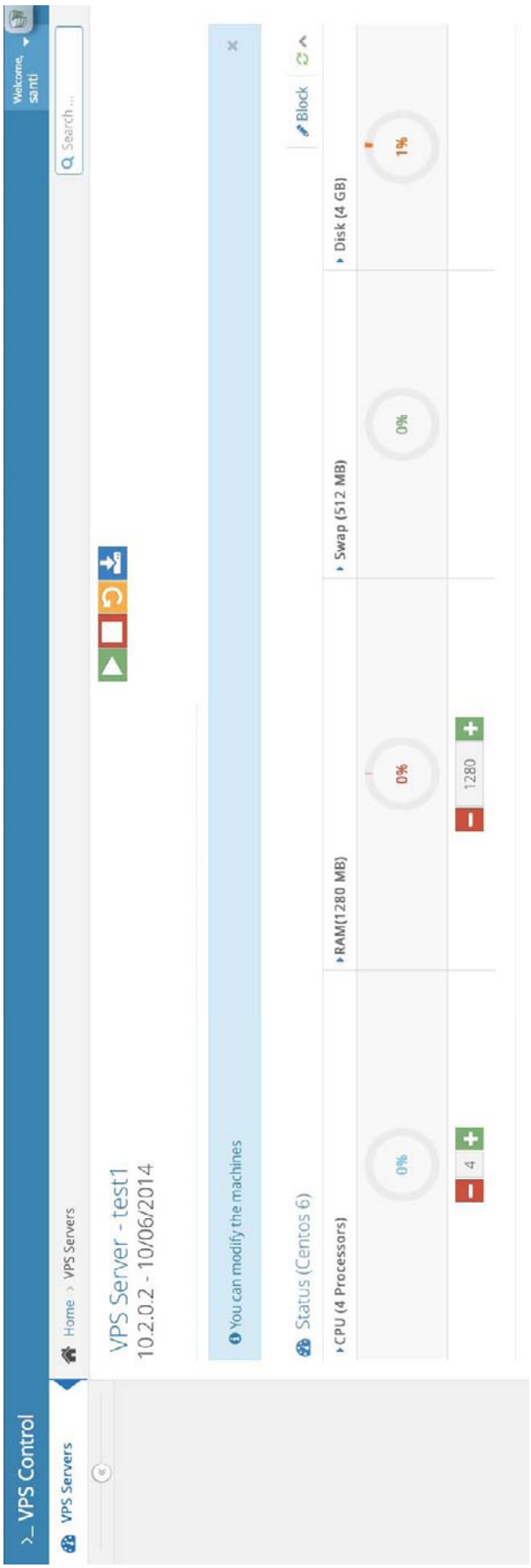


Figura 26: Interfaz con el botón de modificar recursos habilitado.

Cuando se aumenta o se disminuye el número de procesadores o la memoria RAM estos cambios se reflejarán automáticamente en la interfaz así como en el rendimiento de la máquina. Ya que si una máquina trabaja al 50 % con dos procesadores, si se dobla en número (cuatro procesadores) se verá que pasará a trabajar al 25%.

Además, al lado del botón de "Unblock" encontramos otro botón que sirve para refrescar los valores de la interfaz por los valores actuales. El siguiente elemento es un tipo flecha apuntando hacia arriba, que sirve para ocultar la tabla.

Figura 27: Botón de activar la edición de la máquina y botón de refrescar la interfaz.

El otro elemento que queda por explicar en la interfaz, es otra tabla. Esta tabla contiene diferentes gráficas sobre el rendimiento de diferentes componentes durante periodos distintos de tiempo. Estos períodos pueden ser:

- La última hora.
- La última semana.
- El último mes.
- El último año.

Figura 28: Gráficas del rendimiento de una máquina virtual.

En la pantalla anterior se puede observar que se trata del rendimiento de la última hora. En cuanto a las gráficas de tráfico de red y de escritura y lectura de disco, no se observa ningún valor, ya que durante la última hora ningún usuario ha accedido a esta máquina virtual.

Para concluir con las distintas interfaces de nuestra aplicación, nos faltan las pantallas de gestión del perfil de usuario.

Por un lado esta la pantalla (Figura 29) donde te muestra el nombre de usuario, el correo electrónico, la fecha de creación, etc.

Figura 29: Interfaz con lo datos del usuario.

Por otro lado esta la interfaz en la cual el usuario, puede cambiar su usuario, su contraseña o su correo electrónico.

Hay que añadir, que desde esta interfaz un usuario también puede borrar su cuenta de la aplicación si no desea utilizarla más en un futuro.

Figura 30: Interfaz para editar los datos del usuario.

6.2 Diseño del sistema

Dentro del diseño del sistema conviene diferenciar entre el diseño de los procesos y los diagramas de clases. De esta forma la información es más fácil de explicar, y por otra parte, al lector le es más fácil de entender.

6.2.1 Diseño de los procesos

El siguiente punto está dedicado a mostrar los procesos diseñados para las funcionalidades de la aplicación. Para ello se utilizarán diagramas de actividad con tal de ofrecer una mejor comprensión al lector.

En el caso de este proyecto, el proceso principal, y más importante, es el de crear la máquina virtual e instalarle todas las característica que el usuario nos pide. Por lo tanto, este es el proceso que más tiempo de diseño ha requerido. En la figura 31 podemos ver el funcionamiento que se ha diseñado para él.

Otro proceso completo del proyecto, es el de reinstalar una máquina virtual vps, ya que para ello se tiene que borrar completamente la máquina y crearla de nuevo con todas las características iguales que la anterior. El diagrama de actividad sería el de la figura 32.

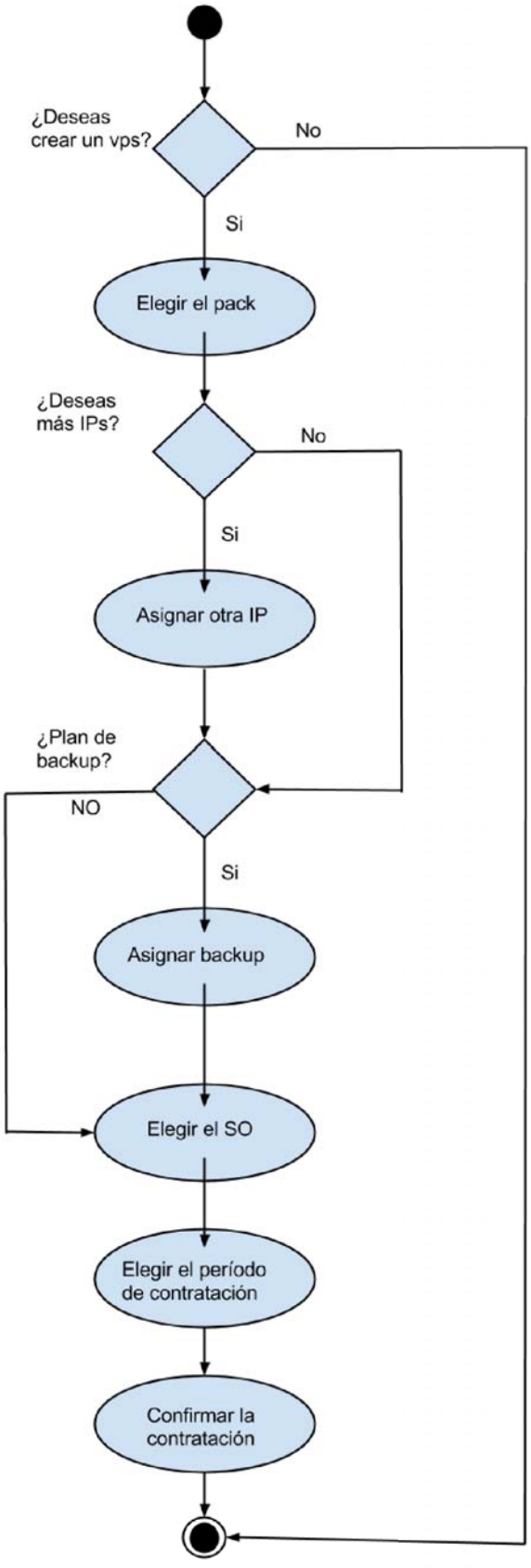


Figura 31: Diagrama de actividad del proceso de crear una máquina virtual vps.

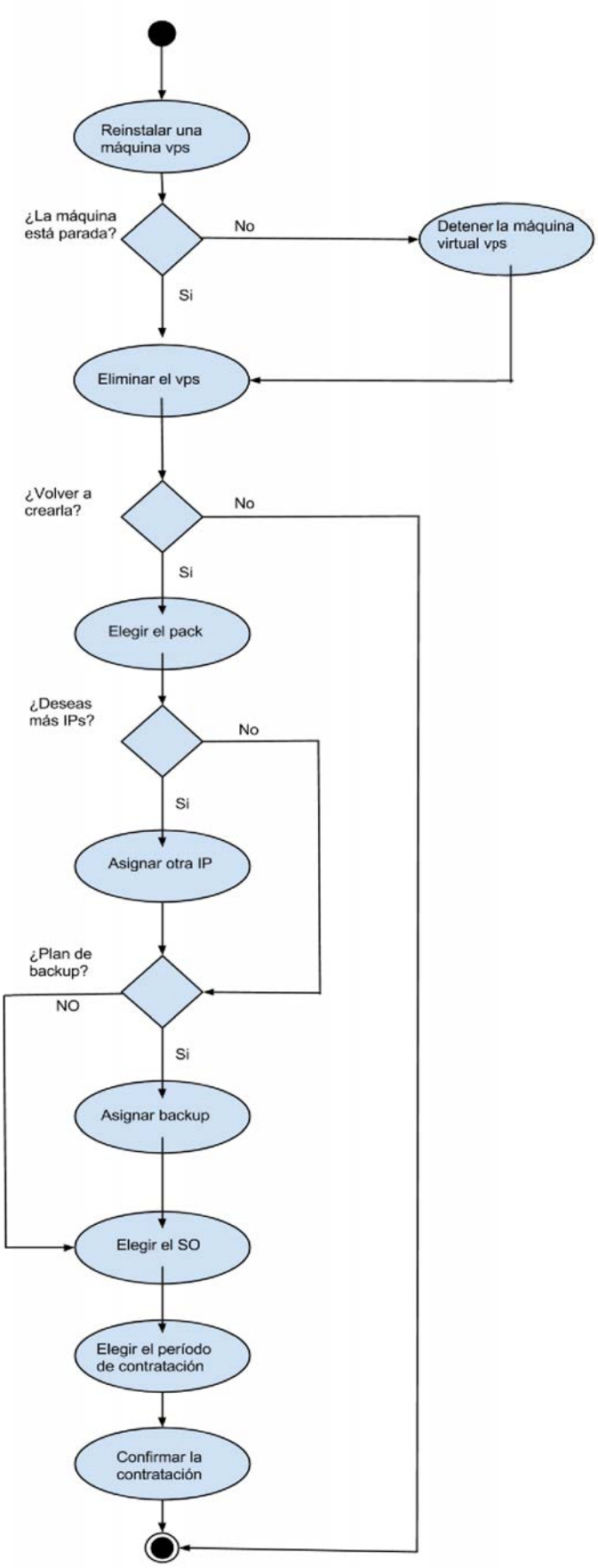


Figura 32: Diagrama de actividades del proceso de reinstalar una máquina virtual vps.

Por último, se encuentra el diagrama de actividad de acceder a una máquina virtual. Este diagrama es importante porque esa es la única forma que tiene el cliente para acceder a la máquina.

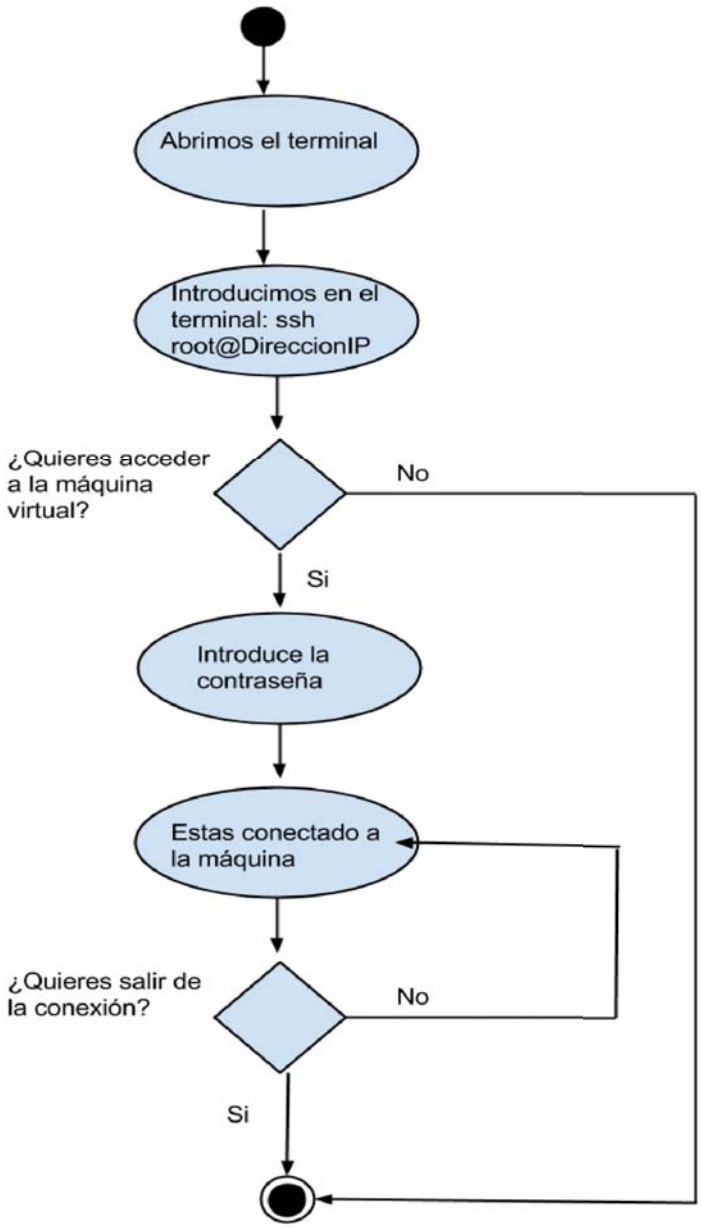


Figura 33: Diagrama de actividades. Proceso de acceder a una máquina virtual mediante open ssh.

Como se observa en la ilustración anterior, la forma de acceder a la máquina virtual es mediante una conexión remota utilizando SSH. Para realizar la conexión tenemos que utilizar una herramienta, en nuestro caso utilizamos open ssh. Para establecer la conexión solo tenemos que introducir en el terminal de nuestra máquina el siguiente comando: *ssh* usuario@ip. Dependiendo de la máquina a la que queremos acceder tendremos que cambiar el campo *usuario* por el nombre del usuario y el campo *ip* por la dirección ip de la máquina que queremos acceder.

Una vez introducido el comando anterior, nos saldrá un mensaje para confirmar las intenciones de

acceder a la máquina. Una vez las confirmamos se nos solicitará una contraseña (la contraseña del usuario con el que hemos decidido conectarnos a la máquina), y si la contraseña es correcta, ya estaremos conectados a la máquina virtual mediante ssh.

En nuestra aplicación siempre que se crea una máquina virtual, se crea un usuario root que tiene todos los permisos sobre la máquina. Además se le asigna una o varias ips. Esos datos son lo que se envían al usuario junto con la contraseña de root cuando se crea la máquina para que pueda acceder y tomar el control.

El resto de procesos que tenemos que llevar a cabo, engloban tareas muy pequeñas, para las cuales no hace falta un diagrama de actividad que muestre su funcionamiento, ya que no responden a más de un par de pasos para ser realizadas.

6.2.2 Diagramas de Clases

En este apartado se muestran los diagramas de clases, de los patrones de diseño utilizados. Los diagramas de clase muestran la estructura del sistema, especificando las clases que este contiene y mostrando los detalles de esta para su implementación.

La siguiente figura muestra gráficamente las clases y los datos utilizados. Seguidamente se explican las más importantes y se describen los atributos:

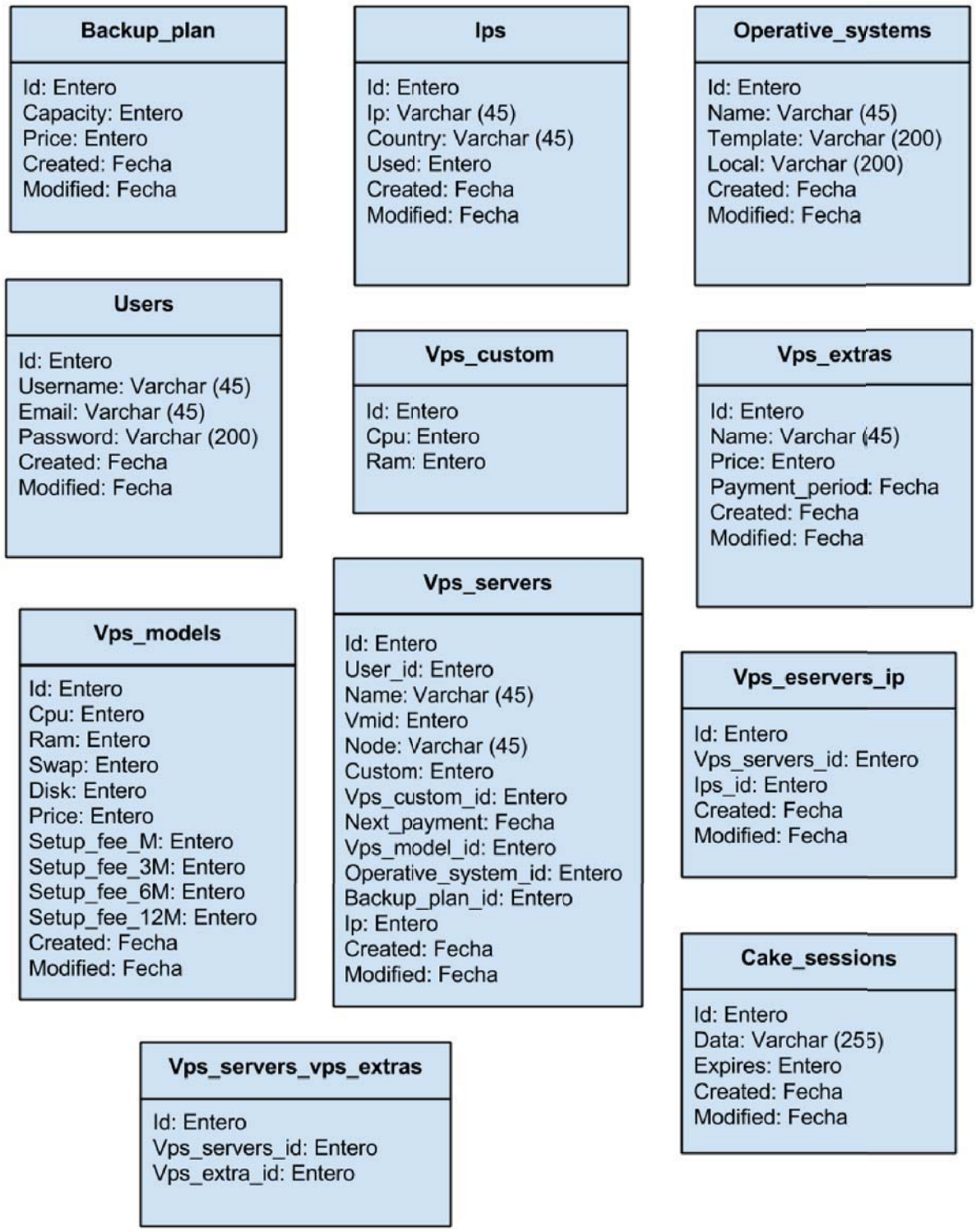


Figura 34: Diagrama clases.

Vps_models: Esta clase contiene la características de las máquinas virtuales que se eligen por defecto, es decir, aquí se almacena cada uno de los diferentes modelos o packs que se ofrecen para contratar una máquina virtual.

- Id: Es el identificador de la tabla (llave primaria). Se trata de un campo de tipo entero.
- Cpu: Es un campo de tipo entero donde se almacenan el número de procesadores.

- Ram: Es un campo de tipo entero donde se almacenan los Megabytes de memoria Ram.
- Swap: Es un campo de tipo entero donde guarda el tamaño de la memoria de intercambio.
- Disk: Es otro campo entero donde se almacena la capacidad total del disco duro de la máquina virtual.
- Price: Es el precio que tiene ese modelo o pack. Este campo es de tipo entero.
- Setup_fee_M: Es de tipo entero y sirve para decir que la máquina se ha contratado para un mes.
- Setup_fee_3M: Es un campo de tipo entero y sirve para decir que la máquina virtual se ha contratado para tres meses.
- Setup_fee_6M: Es un campo de tipo entero y sirve para decir que la máquina virtual se ha contratado para seis meses.
- Setup_fee_12M: Es un campo de tipo entero y sirve para decir que la máquina virtual se ha contratado para doce meses.
- Created: Es de tipo fecha y sirve para saber cuando se introdujo ese modelo a la base de datos.
- Modified: Es de tipo fecha y sirve para saber cuando se modificó la entrada a la base de datos que hace referencia.

Vps_server: Esta clase contiene toda la información referente a la máquina virtual vps alojada en el servidor.

- Id: Se trata de un campo de tipo entero y sirve de identificador de la tabla (llave primaria).
- User_id: Es un campo de tipo entero y sirve para relacionar la máquina con su propietario.
- Name: Es el nombre con el que el usuario bautiza a la máquina cuando la crea.
- Vmid: Es el identificador que utiliza la plataforma de Proxmox para gestionar las máquinas de todo el panel.
- Node: Es el nodo (servidor) donde se encuentra alojada esa máquina virtual.
- Custom: Es un campo de tipo entero que sirve para saber si se a manipulado alguna característica de la máquina después de su creación. Este campo por defecto tiene el valor cero.
- Vps_custom_id: Campo de tipo entero que relaciona la máquina con la tabla custom.
- Next_payment: Campo de tipo fecha que sirve para saber cuando termina el contrato de la máquina virtual para volverlo a renovar si se quiere continuar disfrutando de ella.
- Vps_model_id: Campo de tipo entero que relaciona la máquina virtual con el modelo de características que se contrató cuando esta se creó por primera vez.
- Operative_system_id: Relaciona la máquina virtual con que sistema operativo tiene instalado. Se trata de un campo de tipo entero.
- Backup_plan_id: Relaciona a la máquina virtual con el plan de *back up* que tiene contratado.
- Ip: Relaciona esta clase con la clase *Ips*. Se trata de un campo de tipo entero.

- Created: Es un campo de tipo fecha y sirve para saber cuando se ha creado esta máquina virtual.
- Modified: Es un campo de tipo fecha que sirve para saber cuando se realizó la última modificación sobre una máquina virtual.

El resto de clases se encuentran descritas en el anexo, ya que se ha optado por mostrar dos de las clases más interesantes para el funcionamiento de la aplicación.

Las estructuras de datos anteriores permiten dar soporte a los requisitos de datos que han sido especificados.

7 Implementación

En este capítulo se definen las herramientas utilizadas y el porqué, se explican los diferentes pasos que se han ido siguiendo, se definen las clases y los objetos utilizados, así como sus métodos y se muestra con detalle aquellos aspectos del código que se consideran relevantes.

Durante todo el proceso el estudiante se ha formado de forma continua haciendo uso de diversos manuales de referencia (Api Proxmox [5], etc.), así como realizando numerosas consultas a diferentes sitios Web (tutoriales, foros de desarrollo, etc.) donde se planteaban soluciones concretas o en la propia documentación de la herramienta.

El primer aspecto a decidir para llevar a cabo la implementación fue que herramientas había que utilizar. Una de las características que teníamos que tener en cuenta es que se tenía que integrar en un navegador, lo cual limitaba parcialmente las tecnologías disponibles.

La intención del estudiante era hacer uso de Java como lenguaje de programación de la aplicación y ayudarse de Spring e Hibernate para hacer las conexiones a la base de datos MySQL. Pero el director del proyecto me recomendó realizar un estudio sobre los lenguajes, plataformas, etc. que podía utilizar para realizar la aplicación.

A continuación se muestra el estudio, el porqué de la elección y como es su implementación en el proyecto.

7.1 Base de datos

Para la elaboración del panel de gestión, resulta necesario el empleo de una base de datos en la que se almacenen datos diversos de la aplicación. Se toma esta decisión debido a que se considera más ventajoso que el empleo de archivos para el almacenamiento de la información puesto que:

- El empleo de archivos produce una ocupación inútil de memoria y un aumento en los tiempos de proceso (al repetirse los mismos controles y operaciones en distintos archivos).
- Empleando directamente archivos se está más a merced de las inconsistencias que estas prácticas presentan, ya que la actualización de los datos cuando estos se encuentran en más de un archivo no se puede realizar de manera simultánea en todos ellos.

En este proyecto se va a emplear el Sistema de Administración de Bases de Datos MySQL. Esta decisión se adopta por los siguientes motivos junto al jefe del proyecto:

- Es necesaria la integridad referencial para que la información esté almacenada con una mayor seguridad frente a borrados accidentales, pero no es necesaria la alta seguridad y gran soporte para estos menesteres que Oracle y Postgre SQL, por ejemplo, brindan.
- En apartados posteriores se explican los motivos del empleo de GNU/Linux como Sistema Operativo. Como consecuencia de esto, se descarta por completo el uso de MS SQL Server y Access que necesitan MS Windows.
- Ante todo se busca una buena relación entre calidad, usabilidad y coste. En este ámbito Oracle por ser un sistema de pago y muy costoso queda descartado. Postgre SQL presenta ante todo la mayor lentitud de todos los sistemas analizados, lo que le otorga una demora al sistema innecesaria dado que no es un sistema que vaya a almacenar cientos de miles de entradas de datos.
- Etc.

7.1.3 MySQL

7.1.3.3 Integración en el proyecto

Como se ha podido observar en el diagrama de clases, es necesario crear varias tablas en la base de datos para que esta pueda almacenar toda la información que necesita la aplicación.

Las tablas de la base de datos se crearon mediante sentencias SQL. La creación de algunas tablas están en las siguientes sentencias de código:

```
/*Crea la tabla backups_plans*/

CREATE TABLE `backup_plans` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `capacity` int(11) DEFAULT '50' COMMENT 'In GB',
  `price` int(11) DEFAULT NULL,
  `created` datetime DEFAULT NULL,
  `modified` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


/*Tabla Ips*/
CREATE TABLE `ips` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ip` varchar(45) NOT NULL,
  `country` varchar(45) DEFAULT NULL,
  `created` datetime DEFAULT NULL,
  `modified` datetime DEFAULT NULL,
  `used` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;


/*Tabla Operative_systems*/

CREATE TABLE `operative_systems` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `template` varchar(255) DEFAULT NULL,
  `created` datetime DEFAULT NULL,
  `modified` datetime DEFAULT NULL,
  `local` varchar(45) NOT NULL DEFAULT 'local:vztmpl/',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura 35: Código para crear las tablas de la base de datos MySQL.

Se puede apreciar en el código de la ilustración anterior, que cada tabla tiene un campo que hace de llave primaria y su valor se va incrementando. Como sabemos, un campo que actúa de llave primaria no puede tener el mismo valor dentro de la misma tabla.

Una vez creadas las tablas, estas se tiene que rellenar de datos, para ello algunas de ellas de rellenan mediante la aplicación (crear un usuario nuevo, una maquina virtual, etc), pero otras tablas como por ejemplo, la de modelos o la de sistemas operativos se tienen que rellenar mediante sentencias SQL ya que estas tablas almacenarán información estable para el funcionamiento de la aplicación.

Un ejemplo de instrucción de código SQL sería la siguiente:

```
INSERT INTO `ips` VALUES (28,'10.1.0.1',NULL,NULL,'2014-04-29 10:30:26',1),(29,'10.1.0.2',NULL,NULL,'2014-04-29 10:31:43',1),
(30,'10.1.0.3',NULL,NULL,'2014-04-30 13:50:02',1),(31,'10.1.0.4',NULL,NULL,'2014-05-05 10:53:43',1),(32,'10.1.0.5',NULL,NULL,'2014-05-05 10:54:17',1),
(33,'10.1.0.6',NULL,NULL,'2014-05-05 12:14:47',1),(34,'10.1.0.7',NULL,NULL,'2014-05-05 12:17:46',1),(35,'10.1.0.8',NULL,NULL,'2014-05-05 12:22:30',1),
(36,'10.1.0.9',NULL,NULL,'2014-04-28 13:45:15',0);
```

Figura 36: Instrucción SQL para añadir datos en la tabla Ips.

Como se aprecia, es una instrucción SQL pero inserta varias filas en la tabla Ips.

7.2 Servidor web

Para el desarrollo y funcionamiento de esta aplicación se va a emplear el servidor Apache. Decisión adoptaba por los siguientes motivos:

- Es necesario que el servicio funcione sobre plataforma GNU/Linux, dado que es el Sistema Operativo que tiene instalado el ordenador en el que se va a implantar el sistema. Por esta razón, a pesar de sus servicios complementarios, ISS queda descartado para esta aplicación.
- El sistema requiere estabilidad tanto a nivel de Sistema Operativo como también de la aplicación que trabaja de servidor web, motivo por el que de los servidores web restantes se considera que Apache es el más idóneo por su gran variedad de versiones, antigüedad y amplia comunidad de desarrollo.
- Resulta aconsejable que la aplicación tenga un buen soporte y asistencia técnica, y de todos los servidores web analizados, Apache es el más extendido y utilizado.
- Todo su código está escrito en C, por lo que a pesar de no estar compuesto de las funcionalidades únicamente necesarias, el detrimento que ellas pueden ocasionar en su velocidad no es muy apreciable para el uso que en este proyecto va a tener.
- En algún momento puede ser necesario implementar un sistema de conexiones seguras a través del servidor web, funcionalidad no disponible por parte de Thttpd.

7.2.1 Servidor Apache

7.2.1.1 Integración al proyecto

Para nuestra aplicación tenemos que instalar el servidor apache en el servidor donde se aloja la aplicación con la respectiva base de datos. Posteriormente, tenemos que referenciar la aplicación web a la dirección ip del servidor donde se aloja. Esta tarea la realiza el servidor Apache con tan solo decirle que las peticiones que le llegan las envié a los controladores de la aplicación.

7.3 Proxmox

Proxmox Virtual Environment es una completa plataforma de virtualización basada en sistemas de código abierto que permite la virtualización tanto sobre OpenVZ [11] como KVM [12].

Esta basado en Debian con sólo los servicios básicos para de esta forma obtener un mejor rendimiento.

Proxmox, no es sólo una máquina virtual más, con una interfaz gráfica muy sencilla esta herramienta permite la migración en vivo de máquinas virtuales, clustering de servidores, backups automáticos y conexiones a un NAS/SAN con NFS, etc.

En nuestra aplicación solo crearemos máquinas del tipo OpenVZ. En estas máquinas se puede cambiar tanto la memoria RAM como el espacio en disco asignado, en tiempo real y sin reiniciar el sistema. Otra cosa muy interesante son las plantillas o templates, que consisten en un sistema operativo con algún software preinstalado, que se descarga directamente desde la interfaz de administración y crea una máquina virtual a partir de ellas.

En el proyecto, no se trabajará sobre la interfaz gráfica que proporciona ya que se utiliza la API que tiene desarrollada, pero la interfaz nos es útil a la hora de realizar pruebas y comprobar que las peticiones realizadas sobre la API actúan en consecuencia sobre los servidores.

Todo lo que se puede hacer con el panel, también se puede hacer con peticiones a la API. El primer paso es autentificar la aplicación en Proxmox para que cuando un usuario realice una petición sobre la aplicación, el efecto se vea propagado en Proxmox, y por lo tanto, en las máquinas virtuales de los servidores.

Una vez autentificados solo tenemos que ir realizando peticiones (post, get, put o delete) sobre el API indicando el nodo donde se encuentra la máquina (servidor físico donde se encuentra creada la máquina), el vmid (es el identificador único que asigna Proxmox cuando crea una máquina virtual) y la operación que se desea realizar. Si a la operación se le tienen que pasar atributos, estos también se enviaran junto a la petición.

Como todos los usuarios de la aplicación realizan las peticiones a Proxmox bajo el mismo usuario de Proxmox. Nos dimos cuenta que cuando se realizaban muchas peticiones (100 peticiones/segundo), Proxmox dejaba de responder y algunas de ellas se perdían.

Entonces, después de debatirlo en una reunión con los responsables, lleguemos a la conclusión que lo mejor sería implementar una cola de tareas donde se van guardando todas las peticiones de los usuarios. Por otro lado, habría que crear un demonio, que su misión fuera ir sacando las peticiones de la tabla y realizando las peticiones a la API de Proxmox.

Con esta solución, solucionamos el problema de la pérdida de peticiones, y el tiempo que tenían que esperar los usuarios de la aplicación a que se llevará a cabo su petición era prácticamente el mismo que si no existiera la cola de tareas pendientes.

7.4 Lenguajes Interpretados

Los motivos por los cuales decidimos utilizar PHP para esta aplicación, se describen a continuación:

- Se trata de un lenguaje del lado del servidor, y como hemos mencionado varias veces, esté proyecto tiene una parte al lado del cliente y otra al lado del servidor.
- Al tener que utilizar la API de Proxmox, esta solo puede ser programada en ciertos lenguajes, y uno de ellos es PHP.
- El servidor en el que se va a alojar la aplicación cuenta con Sistema Operativo Debian GNU/Linux y servidor web Apache. Este hecho hace muy complicado utilizar otros lenguajes como, por ejemplo, ASP (Action Server Pages).
- Se persigue una fácil integración entre código y HTML, y PHP permite ser embebido en una

página HTML de una manera muy fácil, y permitiendo una clara separación de código y datos. Y por último es importante tener en cuenta que PHP es interpretado por un modulo de Apache, lo que supone una menor sobrecarga al no tener que hacer uso de un fork/exec para ejecutar el intérprete.

7.4.1 PHP

7.4.1.1 Integración al proyecto

El proyecto de la aplicación esta estructurado de la siguiente manera:

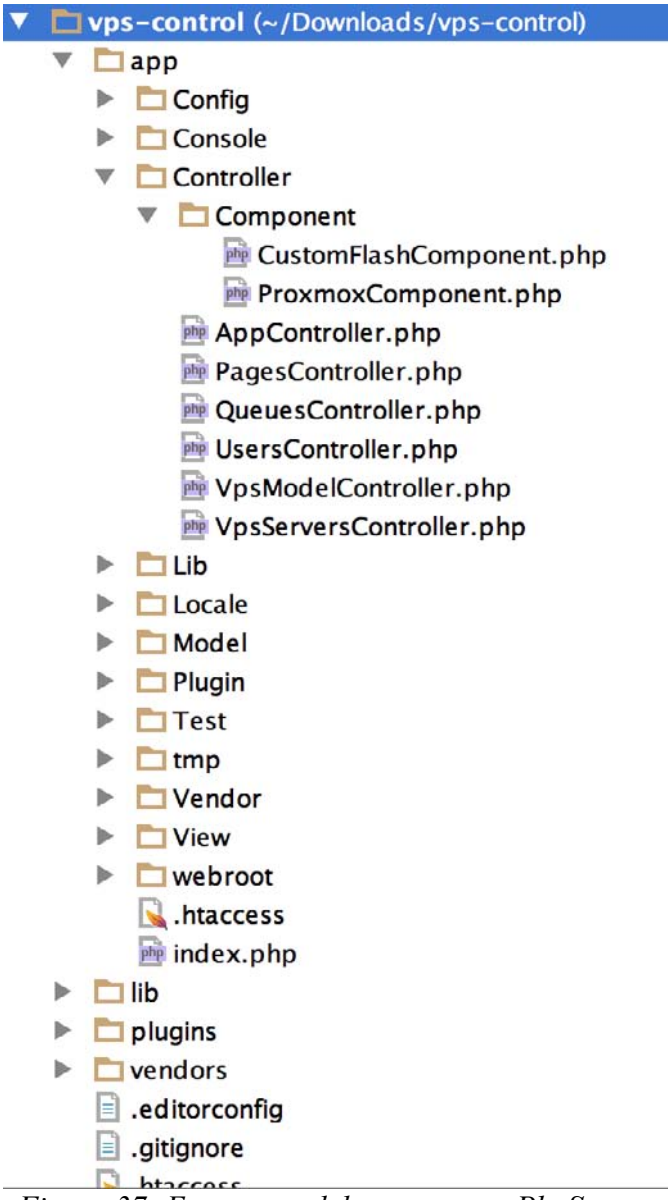


Figura 37: Estructura del proyecto en PhpStrom.

En esta estructura de carpetas se puede observar todos los ficheros .php de los que consta el proyecto. Cada uno de ello se encarga de una función dentro de la aplicación y los ficheros son los

siguientes:

- **Index.php**. Se trata de un fichero especial, ya que es un fichero que contiene el inicio de sesión de la aplicación, pero su extensión es .php ya que es el fichero que carga el servidor Apache cuando recibe una petición y la sesión todavía no está iniciada.
- **AppControler.php**. Como su nombre indica es un controlador que se encarga de gestionar toda la aplicación. Dicho de otra forma, se encarga de controlar todas las características globales de la aplicación. De este controlador cabe destacar que extiende a la clase *Controller* y la función más importante que tiene es la siguiente.

```
public function beforeFilter()
{
    //Pasamos el tipo de usuario a la vista
    $this->set('admin', $this->Session->read('Auth.User.admin'));

    if ($this->Session->read('Auth.User.admin') == 0) {
        if ($this->request->params['controller'] == 'users') {
            switch ($this->request->params['action']) {
                case 'edit':
                    if ($this->request->params['pass'][0] == $this->Session->read('Auth.User.id')) {
                        break;
                    }
                case 'index':
                case 'add':
                case 'delete':
                    $this->redirect(array('controller' => 'VpsServers', 'action' => 'index'));
                //@TODO: Se tiene que implementar que no puedan acceder a una URL directamente
            }

        }
        if ($this->request->params['controller'] == 'VpsServers') {
            switch ($this->request->params['action']) {
                case 'view':
                {
                    $server = $this->VpsServer->find('all', array(
                        'conditions' => array(
                            'VpsServer.user_id' => $this->Session->read('Auth.User.id'),
                            'VpsServer.id' => $this->request->params['pass'][0]
                        )));
                    if (count($server) != 0) {
                        break;
                    }
                }
                case 'extra':
                {
                    $server = $this->VpsServer->find('all', array(
                        'conditions' => array(
                            'VpsServer.user_id' => $this->Session->read('Auth.User.id'),
                            'VpsServer.id' => $this->request->params['pass'][0]
                        )));
                    if (count($server) != 0) {
                        break;
                    }
```

Figura 38: Fragmento incompleto de la función beforeFilter().

Esta función se encarga de dirigirnos a las distintas interfaces dependiendo de la acción del controlador. Además incorpora un filtro que sirve para limitar a los usuarios el acceso a dichas interfaces, es decir, dependiendo del tipo de usuario permite el acceso a unas ciertas interfaces. Por

ejemplo, en el proyecto queremos que solo el administrador pueda eliminar usuarios de la aplicación. Pues entonces a la vista *delete* del controlador de usuarios solo se podrá acceder si estas registrado como administrador.

- **PagesControler.php.** Este controlador es para mostrar las vistas.
- **QueuesController.php**. Este controlador se encarga de la iteración de la aplicación con la tabla *Cake_sessions*. Desde este controlador se ingresan, modifican y se extraen datos de la tabla.

  Una función de este controlador es la de ingresar a la tabla las peticiones que realizan los usuarios para posteriormente ejecutarlas. La función que se encarga de guardar los datos en la tabla es la que se observa en la figura 39:

```
public function add()
{
    // Para no cargar una vista
    $this->autoRender = false;

    if ($this->Queue->save($this->request->data)) {
        $id_task = $this->Queue->getLastInsertID();
        $this->Session->setFlash(__('The queue has been saved.'));
        echo($id_task);
        return;
        //return json_encode($id_task);
    } else {
        $this->Session->setFlash(__('The queue could not be saved. Please, try again.'));
        return json_encode("Error al guardar");
    }
}
```

Figura 39: Función add(). Añade una fila a la tabla Cake_sessions.

- **UsersController.php**. Este fichero es el controlador de los usuarios. Es el encargado de ingresar, modificar, sacar o borrar los datos de la tabla *Users*.

  De este controlador, un método a destacar sería el de *login*, ya que en él se realiza también la llamada al método que se encarga de realizar el *login* a Proxmox mediante la API. El código del *login* es el siguiente:

```php
public function login()
{

    if ($this->request->is('post')) {
        if ($this->Auth->login()) {

            //Accedemos a Proxmox
            $connect = $this->Proxmox->login();
            if ($connect->data->username != "cake@pve") {
                throw new NotFoundException(__('Invalid login proxmox'));
            }

            return $this->redirect($this->Auth->redirectUrl());
        } else {
            $this->CustomFlash->setFlash(__('Username or password is incorrect'), array('alert' => 'danger', 'icon' => 'warning-sign'));
        }
        // if($this->request->)
    }

    $this->layout = 'login_layout';
}
```

Figura 40: UserController. Método login().

- **VpsModelController.php**. Este es el controlador del modelo. Sólo consta de una función que es la que recoge las características de la máquina y las almacena en la tabla extra de la base de datos. Posteriormente se encarga de realizar una llamada a la API del sistema de cobro para que se haga efectiva la compra del servidor vps.

  El código de esta función es el siguiente:

```php
public function pay($id = null) {

    // No se va a mostrar nada por pantalla, no renderizar vista
    $this->autoRender = false;

    if (!$this->VpsModel->exists($id)) {
        throw new NotFoundException(__('Invalid vps model'));
    }

    $infoFrom = $this->request->data;

    // para obtener el precio del backup y la capacidad
    $this->VpsExtra->recursive = 0;
    $extra = $this->VpsExtra->find('all');

    $capacity = 0;
    $price = 0;
    if(array_key_exists('form-field-checkbox-2', $infoFrom) == true ){
        $capacity += $extra['1']['VpsExtra']['capacity'] ;
        $price += $extra['1']['VpsExtra']['price'] ;
    }
    if(array_key_exists('form-field-checkbox-3', $infoFrom) == true ){
        $capacity += $extra['2']['VpsExtra']['capacity'] ;
        $price += $extra['2']['VpsExtra']['price'] ;
    }
    if(array_key_exists('form-field-checkbox-4', $infoFrom) == true ){
        $capacity += $extra['3']['VpsExtra']['capacity'] ;
        $price += $extra['3']['VpsExtra']['price'] ;
    }
    if(array_key_exists('form-field-checkbox-5', $infoFrom) == true ){
        $capacity += $extra['4']['VpsExtra']['capacity'] ;
        $price += $extra['4']['VpsExtra']['price'] ;
    }

    $model = array(
        'amount' =>$infoFrom['precio'] * 100,
        'additional'=> array(
            'vpsModel' => array(
                'id' => $id,
            ),
            'vpsServer'=> array(
                'name' => $infoFrom['hostname'],
            ),
            'operativeSystem'=> array(
                'id'=> $infoFrom['form-field-radio'],
            ),
            'backup_plans'=>array(
                'capacity'=>$capacity,
                'price'=>$price
            ),
            'infoFrom'=> $infoFrom
            )
    );


    $this->SetPayment->setPaymentData($model);
    $this->redirect('/payment/redirect');

}
```

Figura 41: ModelController. Función pay().

- **VpsServersController.php**. Éste es de los controladores más importantes, porque se encarga de todas las funciones que se realizan sobre cada máquina virtual vps.

  Las funciones más importantes de este controlador serían las de crear una máquina o reinstalar una máquina ya existente, pero esas funciones contienen muchas líneas de código, por eso se ha optado por mostrar la función *start*, que se encarga de encender la máquina virtual. Su código es el siguiente:

```php
public function start($vmid, $reboot = false)
{
    // No se va a mostrar nada por pantalla, no renderizar vista
    $this->autoRender = false;

    $var = $this->Proxmox->startOpenVZ($vmid);

    if ($reboot == false) {
        $this->CustomFlash->setFlashMessage(__('Start the vps server'));
    } else {
        $this->CustomFlash->setFlashMessage(__('Reboot the vps server'));
    }
    return $this->redirect(array('action' => 'index'));

}
```

Figura 42: VpsServerController. Función start().

- **ProxmoxComponent.php**. Se encarga de realizar las peticiones a la API de Proxmox. Las peticiones que puede realizar sobre el API pueden ser de diferentes tipos (post, put, get y delete), pero estas peticiones se tienen que realizar mediante *curl* [7] para que la API de Proxmox reciba las peticiones y las interprete de forma correcta.

  A continuación se muestra como se tiene que inicializar el componente *curl*, y posteriormente se mostrarán las funciones de *login* y de listar todas las máquinas virtuales de la aplicación.

```php
public function initialize()
{
    $this->ch = curl_init();
    curl_setopt($this->ch, CURLOPT_SSL_VERIFYHOST, 0);
    curl_setopt($this->ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($this->ch, CURLOPT_PORT, 8006);
    curl_setopt($this->ch, CURLOPT_RETURNTRANSFER, TRUE);
    //curl_setopt($this->ch, CURLOPT_VERBOSE, TRUE);

    if (!$this->_checkLogin()) {
        $this->login();
    }
}
```

Figura 43: ProxmoxComponent. Inicializar el componente curl.

```php
public function login()
{
    curl_setopt($this->ch, CURLOPT_URL, $this->url . '/access/ticket');

    curl_setopt($this->ch, CURLOPT_POST, TRUE);

    $data = array(
        'username' => $this->username,
        'password' => $this->password
    );

    curl_setopt($this->ch, CURLOPT_POSTFIELDS, http_build_query($data));

    $result = json_decode(curl_exec($this->ch));

    $this->Session->write('Proxmox.CSRFPreventionToken', $result->data->CSRFPreventionToken);
    $this->Session->write('Proxmox.ticket', $result->data->ticket);
    $this->Session->write('Proxmox.expiration', strtotime('+2 hours'));

    return $result;
    //@TODO: devolver algun valor y mostrar un mensaje de error si procede
}
```

Figura 44: ProxmoxComponent. Función de acceder en Proxmox.

```php
public function listOpenVZ()
{
    curl_setopt($this->ch, CURLOPT_URL, $this->url . '/nodes/' . $this->node . '/openvz');
    curl_setopt($this->ch, CURLOPT_POST, FALSE);

    curl_setopt($this->ch, CURLOPT_COOKIE, 'PVEAuthCookie=' . $this->Session->read('Proxmox.ticket'));
    curl_setopt($this->ch, CURLOPT_HTTPHEADER, array('CSRFPreventionToken: ' . $this->Session->read('Proxmox.CSRFPreventionToken')));
    return json_decode(curl_exec($this->ch));
}
```

Figura 45: ProxmoxComponent. Devuelve una lista con las máquinas virtuales vps

Hay que destacar que la información devuelta por la API de Proxmox esta en formato de JSON [8], ya que es un protocolo de intercambio de datos ligeros y fácil de leer para humanos y máquinas.

Un ejemplo de JSON sería el siguiente:

```json
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Figura 46: JSON. Fragmento de código.

Solo el texto escrito pertenece al JSON, el resto de paréntesis y guiones son para facilitar su visualización.

- **CustomFlashComponent.php**. Se trata de un componente de la aplicación que se encarga de mostrar mensajes de lo que va sucediendo en la aplicación en segundo plano.

Aparte de los ficheros explicados anteriormente, en la aplicación encontramos otros ficheros de php. Estos ficheros son los que componen los tres plugins que tiene la aplicación y al ser código de terceros pero adaptados a la aplicación no se describirán en detalle.

Esta es la estructura que adoptan los plugins en el proyecto:

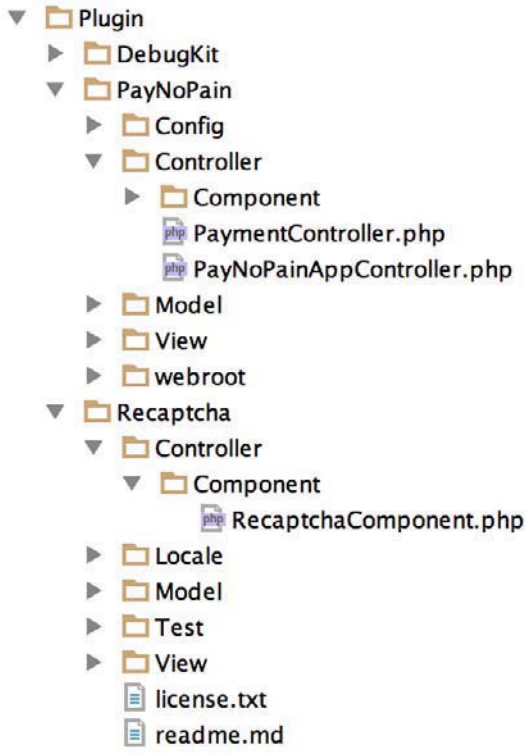


Figura 47: Estructura del proyecto. Parte de los plugins.

7.5 Lenguajes de las interfaces

Las interfaces están implementadas con el lenguaje Html5 [9]. Aunque estas también se apoyan en otros lenguajes como JavaScript, o en librerías como jQuery [10].

La estructura de las vistas del proyecto es la siguiente:

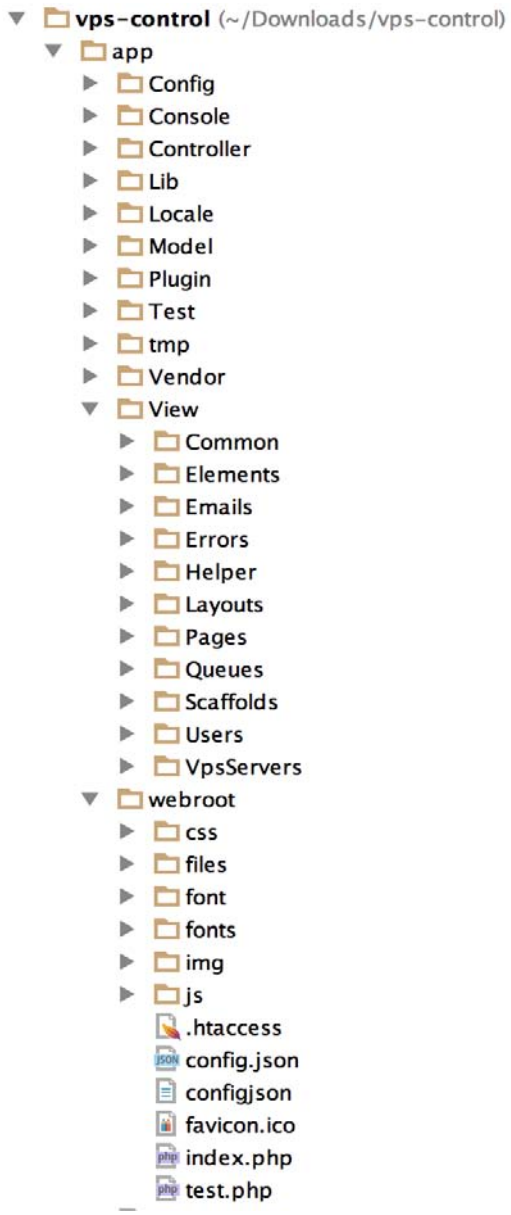


Figura 48: Estructura del proyecto. Vistas, Css y JavaScript.

De la imagen anterior hay que destacar tres carpetas que son:

- **Webroot.** Esta contiene la información para dotar a las interfaces de estilo y de animación. Dentro de esta carpeta destacamos un par de carpetas más:
  - CSS. En esta carpeta se encuentran las plantillas de estilo CSS que se utilizan en el proyecto.
  - JS. Aquí se guardan los archivos JavaScript como las librerías jQuery y los ficheros que contienen scripts. Estos se encargan de interactuar con el usuario y dotar a las interfaces con animaciones.
- **View**. Esta carpeta contiene todos los archivos con extensión *.ctp*. Estos archivos son los que contienen toda la información para generar las interfaces. Se aprecia que los ficheros se encuentran dentro de carpetas con el nombre del respectivo controlador que se encarga de cargarlas. Esta estructura de árbol se usa para facilitar el trabajo a los programadores ya que

las vistas van haciendo llamadas recursivas a los ficheros de nivel más elevado, es decir, a ficheros que cargan los layouts por defecto. Estos ficheros de niveles superiores son comunes para varias vistas.

8. Conclusiones

En este apartado se presentan las conclusiones técnicas y personales obtenidas durante el desarrollo del proyecto.

8.1 Conclusiones técnicas

Tras la finalización del proyecto, desde el punto de vista técnico, se han alcanzado los siguientes objetivos:

- Desarrollo de una aplicación web.
- Construcción de una base de datos en MySQL.
- Gestión de máquinas virtuales en servidores web mediante software libre (Proxmox).
- Gestión de pagos online seguros.

Estas metas alcanzadas suponen que los objetivos planteados por la empresa al inicio del proyecto se han alcanzado. Por lo tanto, se puede afirmar, desde el punto de vista técnico que el proyecto ha sido un éxito.

8.2 Conclusiones Personales

La realización de este proyecto y estancia en prácticas ha sido una tarea muy enriquecedora, ya que se ha obtenido experiencia en diferentes aspectos que no se pueden aprender en clase. Entre los aspectos más importantes, destacaría:

- **Aprender a usar el framework CakePHP**. Este framework no lo conocía antes de realizar la estancia de practicas, pero allí descubrí que es muy útil ya que te facilita muchísimo la faena a la hora de realizar una aplicación Web. Eso es debido a que está compuesto por una serie de herramientas muy útiles, entre las que destacaría:
  - Generación de código.
  - Arquitectura Modelo Vista Controlador (MVC).
  - Ayudantes para AJAX, JavaScript, formularios HTML y más.
  - Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes.
  - Validación integrada.
- **Aprender el lenguaje PHP**. Inicialmente en solitario y luego con el apoyo de los expertos de la empresa, he aprendido los conocimientos necesarios para desarrollar aplicaciones Web.
- **Aprender HTML**. Durante la carrera algo nos enseñaron sobre HTML, pero eran cosas básicas. En este proyecto he aprendido a etiquetar correctamente los elementos de una página web para que luego sea más fácil de introducir JavaScript o AJAX.
- **Aprender JavaScript**. He adquirido conocimientos sobre este lenguaje que resulta muy útil para hacer las interfaces más atractivas para el usuario.
- **Aprender Ajax**. Esto me ha resultado muy cómodo para actualizar contenedores de la pantalla sin tener que recargar la página. De esta forma, la página se recarga antes ya que no

tiene que recargar todos los elementos.

- **Ver las bondades del trabajo en equipo en un entorno real.** A lo largo de la carrera, se han realizado numerosas prácticas de trabajo en equipo, pero durante la realización de practicas en la empresa he podido comprobar los aspectos positivos de este.
- **Dar importancia a las librerías de terceros.** En un proyecto de esta envergadura y con la limitación del tiempo, resulta realmente útil la utilización de librerías estandarizadas que te ahorran el trabajo de volver a implementar algunas funcionalidades sencillas.
- **Conocer la importancia de las pruebas.** He aprendido que un código bien probado por módulos, facilita mucho el trabajo cuando tienes que juntar todos los módulos.
- **Utilización de un servidor real**. Durante la carrera, he trabajado con sistemas que incluían la implementación de la parte correspondiente al servidor, pero no había probado el funcionamiento de esta desde un servidor real.

Ademas de los aspectos descritos anteriormente, también destacaría otros que no tienen nada que ver con la tecnología. Estos aspectos serian la ilusión y las ganas que me ha despertado el proyecto.

El motivo principal fue que a medida que iba desarrollando el proyecto, me iba dando cuenta de la importancia que tendrá en un futuro ya que empresas muy importantes en el mundo de la tecnología están ofreciendo el mismo servicio que pretende ofrecer mi empresa con el proyecto que estaba desarrollando.

Por otra parte, también estaban las ganas de aprender a utilizar el framework de CakePHP o el lenguaje JavaScript, ya que son herramientas que las encuentro muy útiles para el futuro por el mundo en que espero moverme.

Durante el aprendizaje y aplicación de estos aspectos, se han ido viendo conocimientos adquiridos durante la carrera, donde se ha podido comprobar la necesidad y utilidad de algunas asignaturas. De las asignaturas cursadas durante la carrera, destacaría las siguientes para la realización del proyecto:

- Todas las asignaturas de programación, como **Programación Avanzada (EI1017)** o **Estructuras de Datos (EI1013)** donde se aprendieron los conceptos fundamentales sobre esta para la elaboración del proyecto.
- La asignatura de **Sistemas Distribuidos (EI1021)**, donde se adquirieron y aplicaron conocimientos referentes a la arquitectura cliente-servidor, utilizada para la elaboración del proyecto.
- La asignatura de **Base de Datos (EI1020)**, donde se estudiaron los conceptos básico sobre base de datos y su aplicación.
- La asignatura de **Algoritmia (EI1022)**, los conocimientos adquiridos sobre algoritmos y costes computacionales, tuvieron un papel fundamental en la parte de las interfaces, ya que se tuvo que optimizar mucho el JavaScript para que la página se cargará más rápidamente. Además, también nos vino bien a la hora de realizar los algoritmos ya que las llamadas a la API de Proxmox tenían un coste de computación algo elevado.
- La asignatura de **Programación Concurrente y Paralela (EI1024)**, en la que se obtuvieron conocimientos necesarios para la utilización de hilos, utilizados para hacer una cola de tareas y asegurarnos de esa forma que las peticiones a la API de Proxmox se hagan de una en una y no directamente cuando un usuario aprieta un botón.
- Las asignaturas de **Introducción a Redes (EI1015)** y **Diseño y Gestión de Redes (EI1055)**, donde se han adquirido conocimientos sobre la configuración de la tarjeta de red.

- La asignatura de **Seguridad Informática (EI1034)**, donde se han obtenido conocimientos de cómo hacer tu aplicación web más segura, a como almacenar los datos en la base de datos para que usuarios externos no tengan acceso y como hacer copias de seguridad (backup) seguras.

Como conclusión, mencionar que el proyecto me ha servido como repaso de conceptos adquiridos durante la carrera, además de ser una fuente nueva de conocimiento para mi.

Los objetivos que se pretendían conseguir con el proyecto, se cumplieron ya que el panel de control esta completamente terminado a falta de unos retoques con una API para hacer pagos seguros que tiene un pequeño fallo y no realiza correctamente los cobros.

Por eso, en definitiva, estoy muy satisfecho con el trabajo desarrollado en prácticas y con los conocimientos que he adquirido.

Bibliografía

[1]. Sofistic Telematic Security S.L. http://www.sofistic.com/es 2014.
[2]. http://www.computerweekly.com/feature/A-history-of-cloud-computing
[3]. http://www.slideshare.net/cloudbex/acis-charla-cloudcomputing
[4]. https://wrapbootstrap.com/theme/ace-responsive-admin-template-WB0B30DGR
[5]. http://pve.proxmox.com/pve2-api-doc/
[6]. http://es.wikipedia.org/wiki/Secure_Shell
[7]. http://es.wikipedia.org/wiki/CURL
[8]. http://es.wikipedia.org/wiki/JSON
[9]. http://es.wikipedia.org/wiki/HTML5
[10]. http://es.wikipedia.org/wiki/JQuery
[11]. http://es.wikipedia.org/wiki/OpenVZ
[12]. http://es.wikipedia.org/wiki/Kernel-based_Virtual_Machine
[13]. Koulopoulos, Thomas M. (2014): *Navegar en la nube:una nueva forma de pensar acerca del riesgo, la innovación, el crecimiento y el éxito,* Océano.
[14]. Joyanes Aguilar, L. (2012): *Computación en la nube,* S.A. Marcombo.

ANEXOS

Estudio detallado del diagrama de clases

El diagrama de clases, como se ha podido observar en la memoria, es el que aparece a continuación (Figura 49). Y posteriormente se describirán las clases que faltan del diagrama:

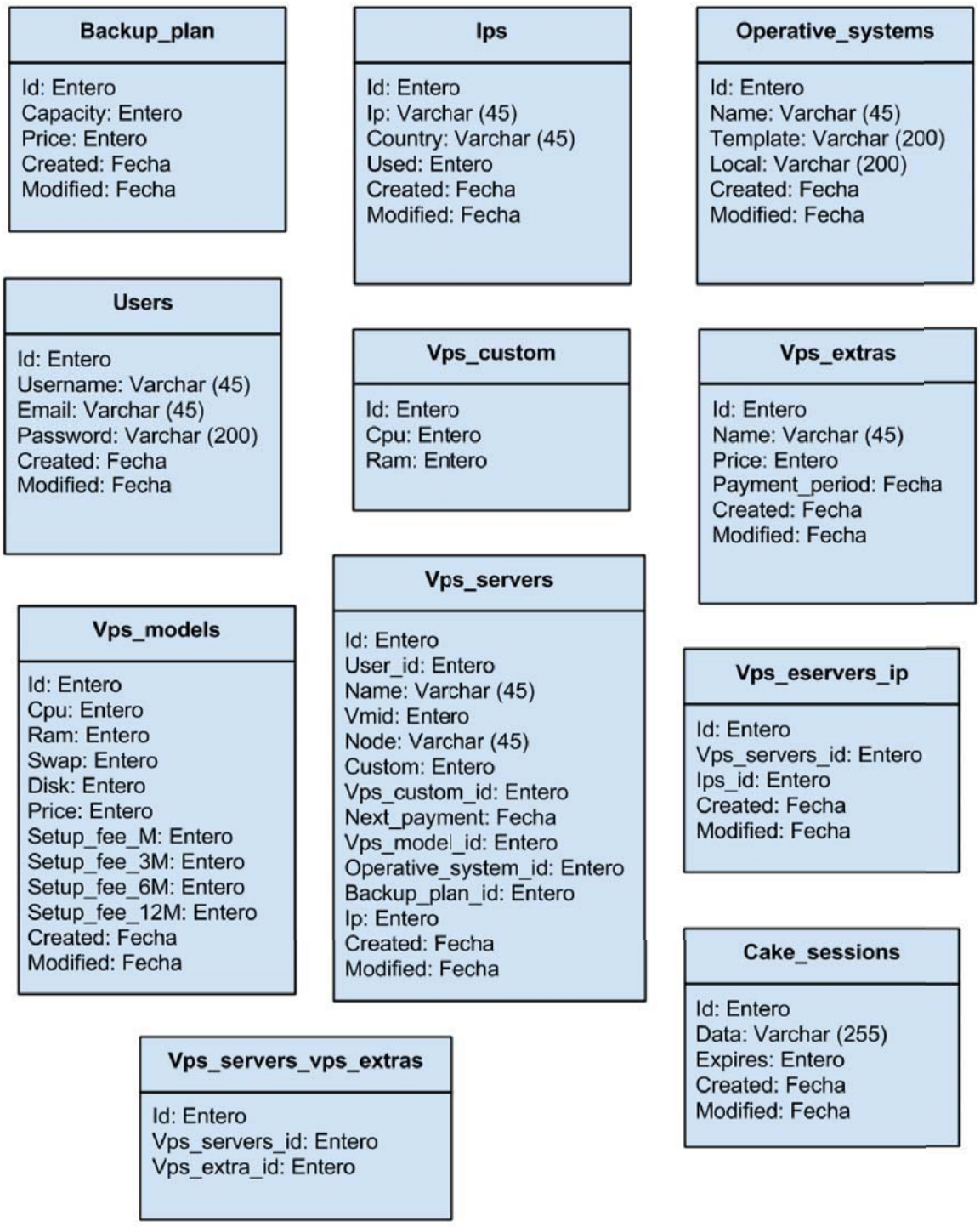


Figura 49: Diagrama clases.

Backup_plan: Esta clase representa cada uno de los planes de *back up* que ofrece la aplicación para las máquinas virtuales.

- Id: Se trata de un entero que sirve para identificar los distintos tipos de planes que hay. Se trata de un atributo único que no puede repetirse. Es necesario para relacionar los planes de *back up* con las máquinas de una forma rápida y cómoda.
- Capacity: Es un atributo de tipo entero y sirve para saber cuanta capacidad de disco ofrece el plan con el que va relacionado.
- Price: Es otro atributo entero que sirve para saber cual es el precio del plan con el que va relacionado.
- Created: Es un atributo de tipo fecha y sirve para saber cuando ha sido introducido ese plan en la base de datos.
- Modified: Es de tipo fecha y su valor es NULL. Si el plan es modificado después de su creación, el campo pasará a tener una fecha mostrando cuando se modificó.

Ips: Esta clase representa las Ips que tiene reservadas o utilizadas la aplicación. Sus campos son los siguientes:

- Id: Se trata de un atributo entero que sirve para identificar a una IP, es decir, es la llave primaria de esta tabla.
- Ip: Es un campo de tipo varchar (45). En el se almacena una dirección IP.
- Country:Es de tipo varchar (45) y se almacena el nombre del país al que pertenece la dirección IP.
- Used: Es de tipo entero y solo puede tomar dos valores, 0 (por defecto) y 1. Se utiliza para identificar si una IP ha sido asignada a un maquina virtual (valor 1) o si esta disponible (valor 0).
- Created: Es de tipo fecha y sirve para saber cuando se introdujo esa dirección IP a la base de datos.
- Modified: Es de tipo fecha y guarda la fecha de la última modificación que sufrió un campo de la tabla.

Operative_systems: Esta clase representa los distintos sistemas operativos que podemos instalarles a una máquina virtual vps. Sus campos son los siguientes:

- Id: Es un campo de tipo entero. Se trata de la llave primaria de la tabla operative_systems. Sirve para identificar una máquina virtual con el sistema operativo que tiene instalado.
- Name: Es el nombre del sistema operativo. Esté campo es de tipo varchar.
- Template: Es el nombre que tiene el sistema operativo. Por ejemplo un valor de este campo seria: “debian-7.0-standard_7.0-2_i386.tar.gz”.
- Local: Es otro campo de tipo varchar y en el se almacena la ruta hasta donde se encuentra el template o sistema operativo a instalar.
- Created: Almacena la fecha de cuando se introdujo un sistema operativo a la base de datos.
- Modified: Es de tipo fecha y almacena la última vez que fue modificada esa línea de la base

de datos.

Users: Esta clase sirve para almacenar toda la información referente a los usuarios de la aplicación. Sus campos son los siguientes:

- Id: Es un campo de tipo entero. Se trata de la llave primaria de la tabla users. Sirve para identificar a los usuarios.
- Username: Es un campo de tipo varchar y es el nombre de usuario para acceder a la aplicación.
- Email: Almacena el correo electrónico del usuario. El campo es de tipo varchar.
- Password: Es un campo de tipo varchar que almacena la contraseña del usuario codificada.
- Created: Almacena la fecha de cuando se registró un usuario a la aplicación.
- Modified: Almacena la fecha de cuando un usuario ha modificado algún dato de su perfil.

Vps_custom: Esta clase sirve para relacionar una máquina con sus características hardware una vez se ha modificado alguna de ellas. Dicho de otra forma, se produce una entrada cuando se modifica una característica hardware (Número de procesadores o CPU).

- Id: Es un campo de tipo entero que sirve para identificar los elementos de la tabla vps_custom (Llave primaria).
- Cpu: Es un campo de tipo entero que guarda el número de procesadores de las máquinas virtuales.
- Ram: Se trata de otro campo numérico. Este campo almacena los MegaBytes de memoria Ram que tiene cada máquina virtual.

Vps_extras: Es una clase para relacionar el periodo de contratación con la máquina virtual. En esta tabla se muestran los precios de cada tipo de *Setup* que se elige.

- Id: Es la llave primaria y sirve para relacionar esta tabla con la tabla intermedia Vps_servers_vps_extras
- Name: Es el nombre que se asigna al tipo de periodo de tiempo para el que se desea contratar la máquina. Este campo es de tipo varchar (45).
- Price: Es de tipo entero y se almacena lo que cuesta la puesta el mantenimiento del servidor para ese periodo concreto de tiempo.
- Payment_period: Identifica cuando se terminara el contrato sobre una máquina virtual contratada. El campo es de tipo fecha.
- Created: Almacena la fecha de cuando se realizó esa entrada a la base de datos.
- Modified: Almacena la fecha de cuando se modifico por última vez un campo de la fila a la que pertenece.

Vps_servers_ips: Se trata de una tabla intermedia para realizar una relación de uno a muchos. Las

tablas que se intentan relacionar son: la clase de vps_servers y la clase ips.

- Id: Es el identificador de la clase. Se trata de un campo de tipo entero y hace de llave primaria dentro de la tabla.
- Vps_server_id: Se almacena un entero que identifica a la maquina virtual (vps_server).
- Ips_id: Es un campo de tipo entero y en el se almacena el identificador (llave primaria) de la tabla ips.
- Created: Almacena la fecha de cuando se creo una entrada en la tabla.
- Modified:Almacena la fecha de cuando se modificó una entrada de la tabla.

Vps_servers_vps_extras: Es una clase intermedia para realizar una relación de uno a muchos entre dos tablas (la tabla vps_servers y la tabla vps_extras).

- Id: Es la llave primaria de la tabla. Se trata de un campo de tipo entero.
- Vps_server_id: Es un campo de tipo entero donde el identificador de una máquina almacena en la tabla vps_server.
- Vps_extra_id: Es un campo de tipo entero donde se almacena la llave primaria de la tabla vps_extra.

cake_sessions: Esta clase no tiene relación con las anteriores ya que sirve para almacenar tareas pendientes y las tareas ya tratadas. Dicho de otro modo, como todos los usuarios no pueden realizar peticiones a la vez a Proxmox ya que de hacerlo se bloquea la plataforma, se ha decidido crear una tabla donde se almacena cada petición de los usuarios. Y hay un demonio que esta leyendo la tabla y cuando se encuentra una tarea nueva la ejecuta. De esta forma, las peticiones a la Proxmox se ejecutan una detrás de otra.

- Id: Es el identificador de la tarea. Es la llave primaria de la tabla.
- Data: Es un campo de tipo varchar (255) y sirve para almacenar la petición que se tiene que lanzar contra Proxmox.
- Expires: Es un campo numérico que puede tener tres estados (cero, uno o dos). Si el campo se encuentra en cero significa que la tarea todavía no ha sido tratada, es decir, termina de entrar a la base de datos. Si tiene el valor uno, significa que se esta tratando en ese instante. Por último, si obtiene el valor dos significa que la tarea ya se ha realizado.
- Created: Es un campo de tipo fecha que sirve para saber cuando se mandó a ejecutar la tarea.
- Modified: Es un campo de tipo fecha y almacena la fecha de cuando se modificó por última vez. Este campo sirve para saber cuánto se ha tenido que esperar un usuario desde que ordenó un cambio hasta que este se ha efectuado.